Lithium-Ion Battery Data Management

Frank Ferrato

Dr. Jung-Hyun Kim

April 2018

Abstract:

Lithium Ion Battery research is growing due to the need for renewable resources. Since the amount of research is growing so is the amount of data collected. With all this data it is taking more and more of researcher's time to process all the raw data collected by battery cell cyclers, such as Arbin. Dr. Jung-Hyun Kim, my research advisor, has estimated that it takes up about a third of his time right now just to process data after running tests. With this rise in research the goal is to automate this process, by creating an open source universal user-friendly program. Initially it starts with understanding all the ways that researchers will analyze data to find different properties such as, capacity and cycle life of a battery. Alongside of this an understanding of how raw data is produced and what format it comes out in. With both of these locked down a beginning program to process the data can be designed. To make it user friendly a GUI (Graphic User Interface) will likely be used so that instead of someone knowing how to code all they have to do is understand what kind of data they want to process and take a few clicks to get the final process of the data.

Once a prototype program is created it will be shared with many of Dr. Kim's researchers for them to use with their own data that are collecting for them to characterize their battery cells. Feedback about the program will be obtained and improvements will be made. Then this program will be shared on the internet to create an open source base program. The hope is that many universities will use this to conduct their own analysis. Not only will they use it but they will make improvements on it and continue to share their improvements to the program and addons to better the research community of Lithium Ion Batteries. This could be the beginning of a program that will change the way battery research in conducted hopefully cutting the time that Dr. Kim has estimated from 1/3 to less that 1/20 of a researcher's time. Put the struggle of processing data behind them to allow them to focus more time on improving batteries in safety capacity and cycle life.

Acknowledgements:

I would like to thank Dr. Jung-Hyun Kim, not only was he accepting to having me on his team but also several others, he recognized my interests and gave me a project that fit them well. He pushed me to do my very best work every week and to always be prepared. He became a mentor and to me and others in his lab. He was always friendly and willing to help with any questions you had. So, thank you Dr. Kim. I would also like to thank Cody O'Meara for helping me with my project as well. He got me in my feet in the beginning and gave me a basis for Lithium-Ion batteries and made everything such a seamless transition. Not only was he able to get me on my feet in running but also provide meaningful feedback throughout the entire process and remain engaged while keeping track of his own project. He was also available whenever I needed him and helped me with the final stretch of my project when it came to crunch time. Once again thank you to Cody.

Table of Contents:

List of Tables:

List of Figures:

1: Introduction

Lithium-ion (Li-ion) batteries are used everywhere in today's world, including nearly every cell phone and laptop, with new uses in electric vehicles and grid energy storage currently being commercialized. They are continuously being improved to have larger capacities and enhanced safety. These improvements allow for the increased use of renewable energy sources instead of using non-renewable energy sources, such as fossil fuels, for everyday activities.

A lot of research is being conducted to improve batteries, including at many universities looking to find new technologies to improve battery performance. A great deal of testing is involved to develop new battery technologies, and once the testing is complete, more time is spent to process and analyze the data. Processing and analyzing the data is necessary to determine whether these new developments are feasible or not, and thus is an extremely important aspect of the research process. Some companies provide software to plot the data produced by their hardware, allowing the user to easily select the data/graphs they want to output, ultimately saving time. However, these devices are expensive and are out of the price range of many university researchers. Thus, each individual researcher must complete data-processing on their own, which can be time consuming.

1.1: Research Purpose

Since data processing takes so much time to complete, it is beneficial for someone to be devoted to improving the techniques for the entire group. This data analysis software will be made open source for the use of any group who may have a need to process and analyze their cycling data. The final product should be able to easily produce many common types of characterization plots for batteries. All data should be able to be compared quickly and painlessly to determine exactly the outcomes of testing.

Ultimately, this should provide researchers more time for improving batteries and to cut down on the time it takes to present test results. Much time will be spent waiting on testing to be finished and during this time, instead of crunching data to characterize batteries researchers can work on multiple projects at once and more effectively improve batteries for all people in the world. Since most of this work is preliminary, if data analysis can be done faster, proof of concept can be done quicker when pitching this idea for companies to adopt and test. The community will be able to advance batteries faster and create a revolution quicker than anyone could ever expect.

1.2: Outline of Thesis

This thesis includes sections covering the software development, instructions on the software use, and sample graphs produced by the software. The software development will cover how the data is collected and transferred into the software. It will cover how the data is processed in MATLAB to produce graphs quickly and accurately to compare data. Ultimately, all your data should be accessible through the software so that someone can compare all data that has been collected by the user and create a more comprehensive comparison. The functionality in each tab will also be covered in section four. After a good understanding of the software is developed, a set of instructions for the user will be included. Finally, a set of essential plots produced by the software will be shown. First, the background will touch on the intricacies of a software like the one being developed and Arbin data collection.

2: Background

2.1: Similar Program

Maccor is a battery cycler that comes with its own analysis software included. This program works directly with the data format that the battery cycler produces. It produces well-made graphs and charts that are customizable. Raw data can also be retrieved from it. The largest downfall of this program is that it is purposely designed to only work with the Maccor battery cycler, thus baiting you into to buying the more expensive hardware.

Eventually everything the Maccor software does should be able to be accomplished in a similar manner in the software being developed for Arbin. Another nice feature that is included in the Maccor software is the ability to save a template of the graphs you produced and import this template with any sets of data. Since the Maccor software was probably developed over a much longer time and with much more input and improvements, our software will not have nearly as much functionality as the Maccor software, but it makes a good benchmark for what a program of this nature should reach for.

2.2: Arbin Data

Arbin battery cyclers are affordable and many universities use them to collect data. The biggest problem is their lack of ability to give the user data that is easily understandable and developed. Currently, users of Arbin battery cyclers either develop their own method of data analysis, or, the most time consuming, do data analysis on the fly. If they chose to do the latter, more time is consumed to perform individual analysis every time. The first option also uses a large amount of resources and if you do not have the budget to pay for this, then it is not an option. This leads us to another option, which is to create your own software to automate the data analysis.

The biggest hurdle is being able reliably handle the large amounts of data that come with battery testing. For example, if a battery is tested for 500 cycles it can produce anywhere from

50,000 to 120,000 data points, depending on the settings. If you think about all the channels that can be used and all the comparisons that can be made it very quickly becomes a large amount of data. Until recently, the only format that Arbin would give you raw data in was Excel, and you could only look at one data set at a time. This meant that if you wanted to compare two different data sets you would have to copy one over to the other and produce plots that way.

Since excel data is still the main way Ohio State looks at data, it is essential that we can convert this over to MATLAB to allow for an easier comparison across data sets. Dr. Jung Hyun Kim's group recently purchased an Arbin cycler that allows the user to access an entire Structured Query Language (SQL) database and complete queries that way. SQL as stated is a code language that can work in conjuction with MATLAB to be able to link to the database remotely and access data based off queries. Unfortunately, this product is so new it has not yet been networked, so it is not advantageous to pursue this path yet. Hopefully, when this functionality becomes more available it will be more of a plug and play process and much of the backend of the program will remain the same.

3: Program Basis

The newly developed software is based off an original software developed by Cody O'Meara. His software produces four graphs that where necessary for his data analysis. The new software also produces these and allows the user to put graphs together using a secondary Y-Axis. Along with these four graphs, you can produce many other types of graphs by selecting the X-Axis and Y-Axis independently. Below is the screenshot of the software that O'Meara developed.

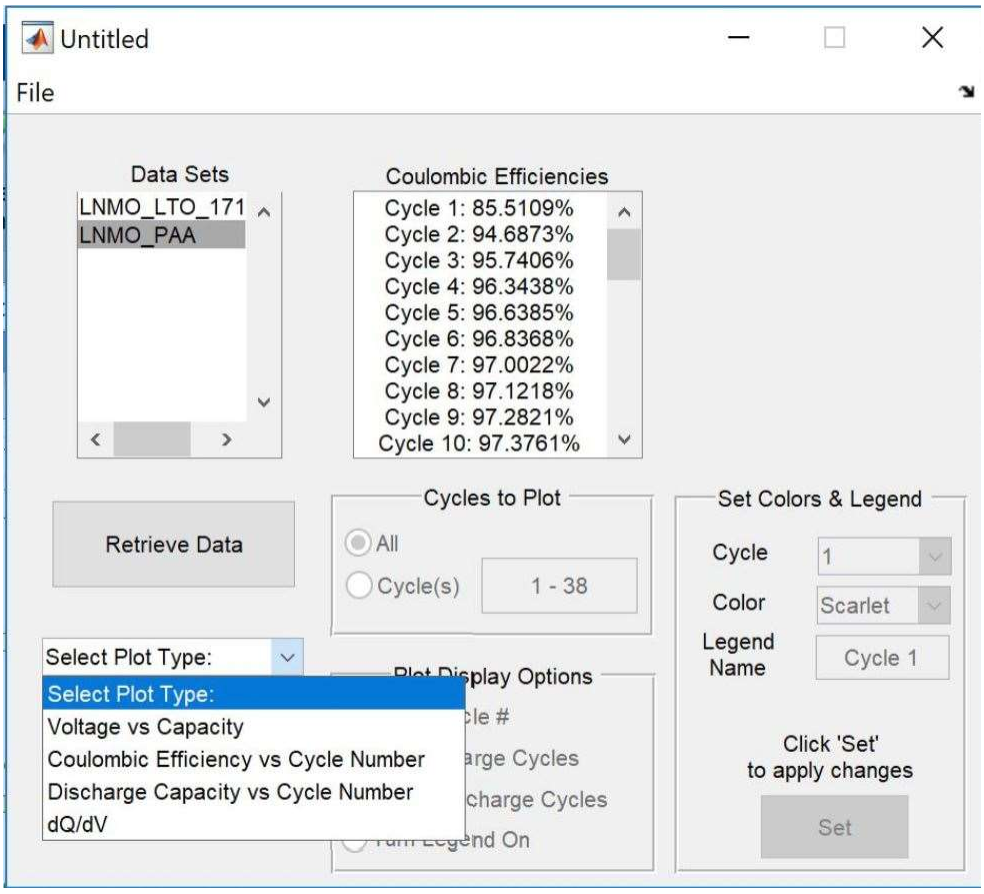

Figure 1: O'Meara's Software User Interface

This software was a good basis and there will be further development of several ideas used in the software. The software starts by asking the user to choose a folder where the data that you are interested in is stored. Once you choose this folder it loads all excel file names, and from

there you can choose one or multiple different data sets to compare. Once that is completed the user retrieves the data, which imports from the excel spreadsheets and then organizes data in MATLAB to help with producing graphs. You can choose what type of plot you want based off the four plots, and then make changes to the legend and other components of the plot. A very similar process was used to allow the user to retrieve the data from the excel spreadsheets. Another functionality that needs to be added is the SQL database retrieval process. If set up correctly, you should be able to use similar plotting functions and change the front end of the program to organize the SQL data in the same manner as the excel data.

The major difference between the program that O'Meara developed and this one is that the user switches between tabs and each tab has a different purpose. The four tabs are Data Selection Tab, Axis Selection Tab, Legend and Plot Tools Tab, and Raw Data Tab. Below I will describe each tab, where their development is, and what can be improved in each section. The development of each tab will cover both the user interface and the purpose each serves in the process. A picture of the user interface will be included in each section as well.

4: Program Interface

4.1: Data Selection Tab

The first tab is the Data Selection Tab, which contains four different sub-panels. The first sub-panel that the user will use is the Data Format panel. In this panel, the user selects the format of the data that they are interested in analyzing. Depending on which format you select, the code will enable or disable the SQL and Excel Data panels.

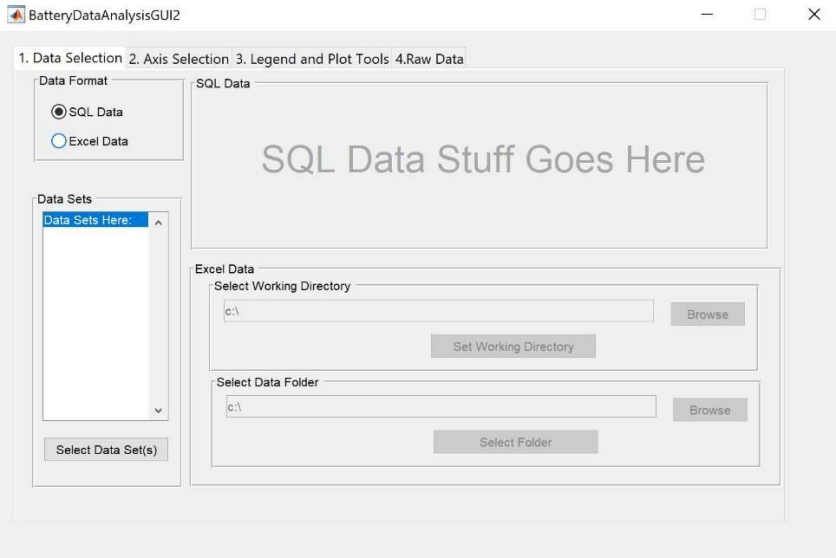

Figure 2: Data Selection Tab

The SQL and Excel Data panels are never both enabled, so the user cannot analyze different data sets across different data formats. Since Dr. Kim's lab just received the battery cycler which produces data in SQL, there has not been enough time to create the interface to get the data from the SQL database directly. Another issue with the SQL is that right now the database is not a part of the network, so the only way to access this data is to be on the exact computer that houses the data. Once the database becomes networked, it will be more beneficial to be able to connect to it and retrieve data directly from it. The Excel Data panel, shown in *Figure 3,* allows the user to pick a working directory and a folder that has all the data you are interested in. When you select the folder, it creates a list in the Data Sets list box (*Figure 4*) to allow the user to select the data sets.

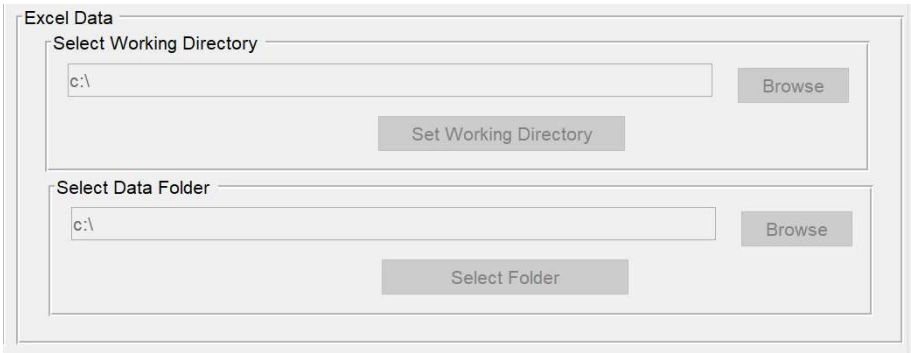

Figure 3: Excel Data Panel

Once the user selects the data sets they are interested in they click the “Select Data Sets” button which processes all the raw forms of data and saves them in the background workspace. This takes a rather large amount of time, especially with larger data sets in the hundreds of cycles. To reduce this, it has been proposed to remove any sorting of data and calculations performed here. These would be moved to the axis selection which would extend the time of in that tab but reduce the overall by only performing calculations after the user decides they need them. Once the data is retrieved the user will switch to the Axis Selection Tab.

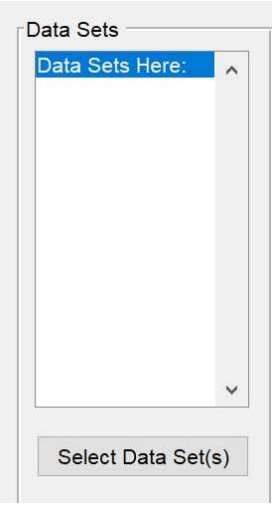

Figure 4: Data Sets Panel

4.2: Axis Selection Tab

In the Axis Selection Tab, the user will go through and make choices for what kind of data they are interested in plotting. This allows the user to quickly see trends in their battery cells that would otherwise take large amounts of time to sort through data and compare. The user will choose separately what they want to plot on the X-Axis, Y-Axis, and Secondary Y-Axis (right side).

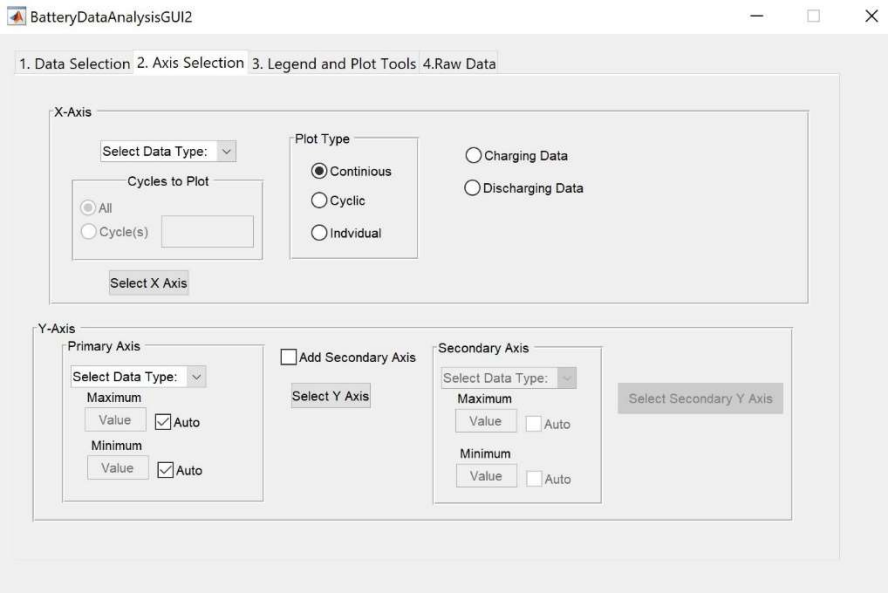

Figure 5: Axis Selection Tab

First the user will select the X-Axis data type. After the user selects this, depending on what type of data they select, options for other boxes will enable or remain disabled. These boxes include Cycles to Plot, Charging and Discharging Data, Plot Type options. The cycles to plot box allows the user to choose to plot all cycles or choose to plot only specific cycles. If the user selects either Charging or Discharging Data or both it will give them the respective data on their plots. By choosing a continuous plot, the time series data will continue to grow. By choosing cyclic or individual, the time series data will reset for each step Charging or Discharging. After the selections are made, the user must click “Select X-Axis”. *Figure 6* below shows a closer view of the X-Axis Selection.

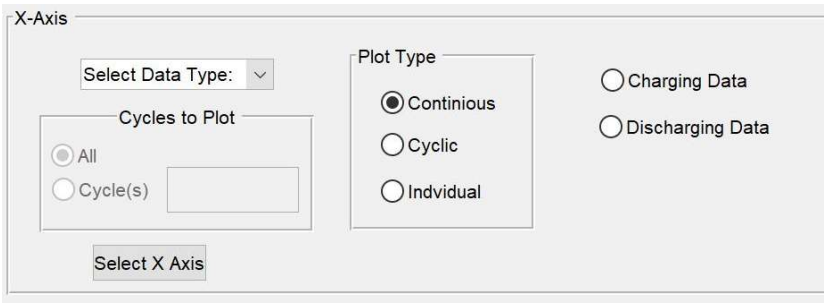

Figure 6: X-Axis Selection Panel

The user will then move on to Y-Axis data type selection. This includes the selection of up to two types of data that will be plotted with the X-Axis data type that was chosen above. Along with each, the user will have the option to select the range of the Y-Axis, or leave it as automatic. They can also choose these same options on the secondary Y-Axis. For both the user must finally click “Select (Secondary) Y Axis” as shown in *Figure 7*.

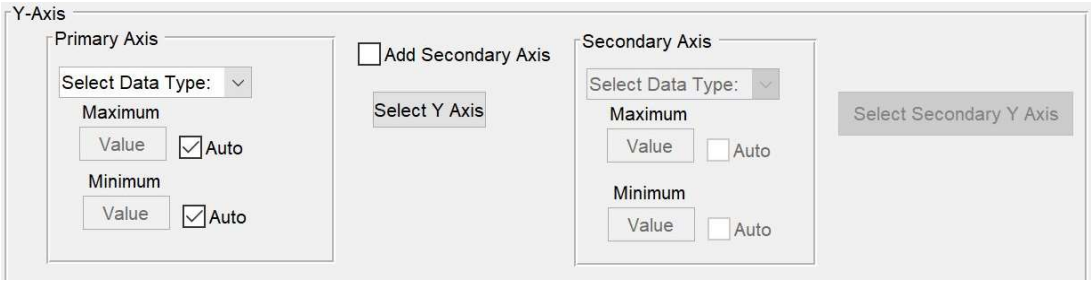

Figure 7: Y-Axis Selection Panel

4.3: Legend and Plot Tools Tab

The third tab allows the user to personalize the plot to their liking. If you leave “default all” it will plot all Y-Axis data as blue and all Secondary Y-Axis data as red, and will have no markers or legend included with the plot. By unselecting “default all” the user can personalize the figures.

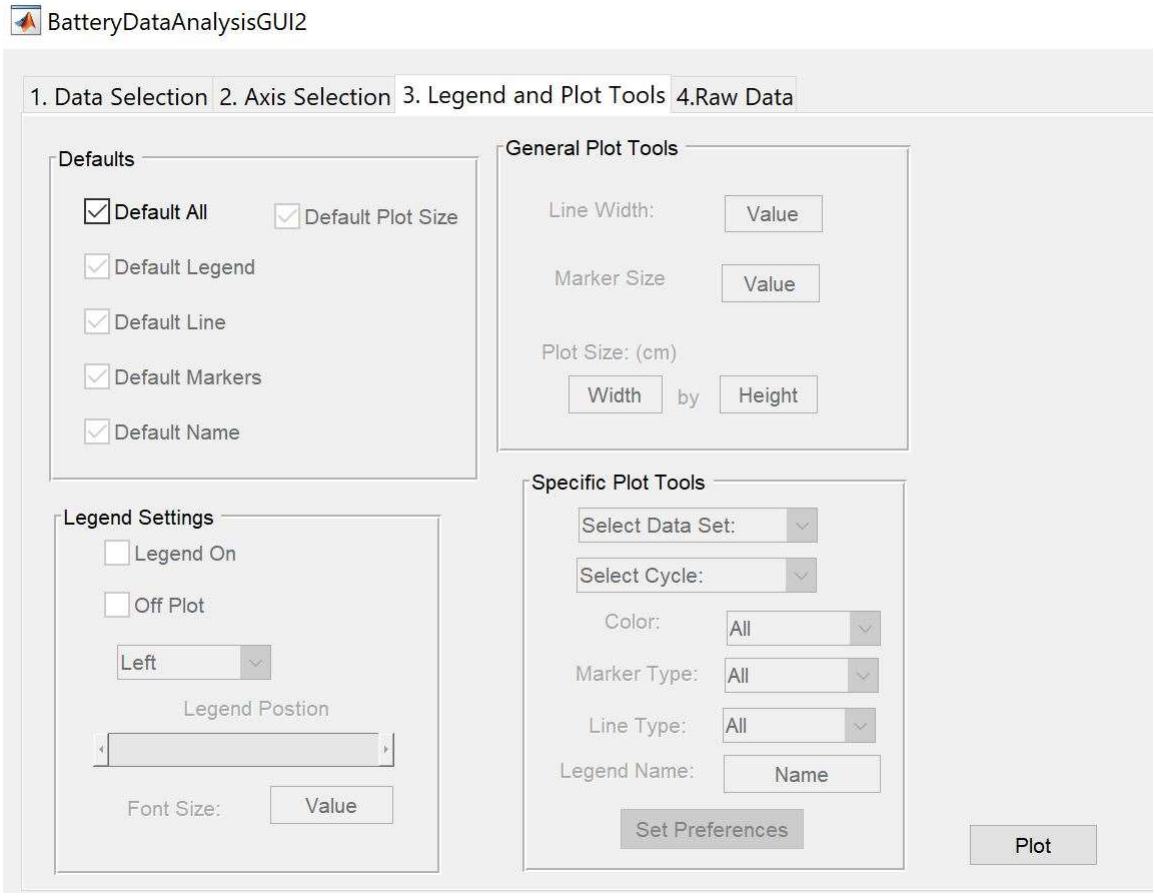

Figure 8: Legend and Plot Tools Tab

By then deselecting the legend, it opens the Legend Settings Panel and Specific Plot Tools, except for the Legend Name box. This allows the user to turn the legend on, choose to have it on or off the plot, what side of the plot, and where on that side. For example, if you choose to put the legend on the right side and slide the bar all the way to the right, it will put the legend in the top right corner. They can also choose the font size of the legend when it is produced. In the specific plot tools panel, (*Figure 9*) the user can pick a specific data set and cycle to set options of that cycle from line type, color, marker type. If they also deselect the name option they can change the name that will appear for that cycle in the legend.

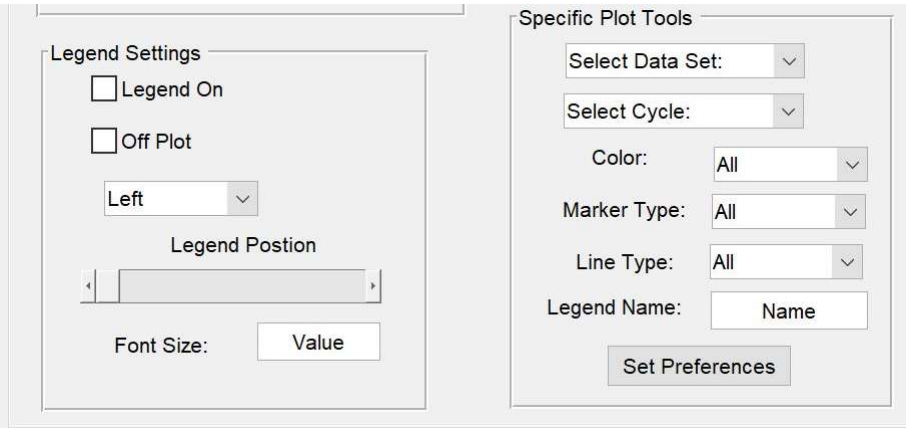

Figure 9: Legend Settings Specific Plot Tools

Finally, you can deselect Marker Line and Plot Size which allows the user to change the size of the markers, size of lines and the overall size of plot. These are all under the general plot tools (*Figure 10*) it will be a setting that covers the general idea of what the plot will look like.

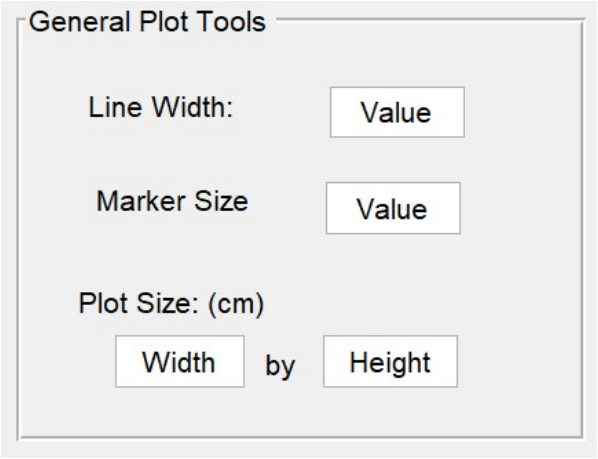

Figure 10: General Plot Tools

The nice thing about this portion is the user can choose to customize as much of the plot as they want, or choose to not customize it at all.

4.4: Raw Data Tab

After the user plots their figure(s), they have the option to export the data in the Raw Data Tab. They do this by selecting the Data Set they want to export the data for. *Table 1* below shows the form in which the data will be inputted in the table.

Table 1: Sample Raw Data Format

X-Axis Charging Cycle XX	Y-Axis Charging Cycle XX	Secondary Y-Axis Charging Cycle XX	X-Axis Discharging Cycle XX	Y-Axis Discharging Cycle XX	Secondary Y-Axis Discharging Cycle XX	… Repeat Cycle YY
Selected Data Type (ex. Time)	Selected Data Type (ex. Current)	Selected Data Type (ex. Voltage)	Selected Data Type (ex. Time)	Selected Data Type (ex. Current)	Selected Data Type (ex. Voltage)	… Repeat

The user will also be able to see the Columbic Efficiency for each cycle, which is calculated by dividing the energy you get out of the cell by the energy put into it. All of this will show up once the user selects “Get Raw Data”.

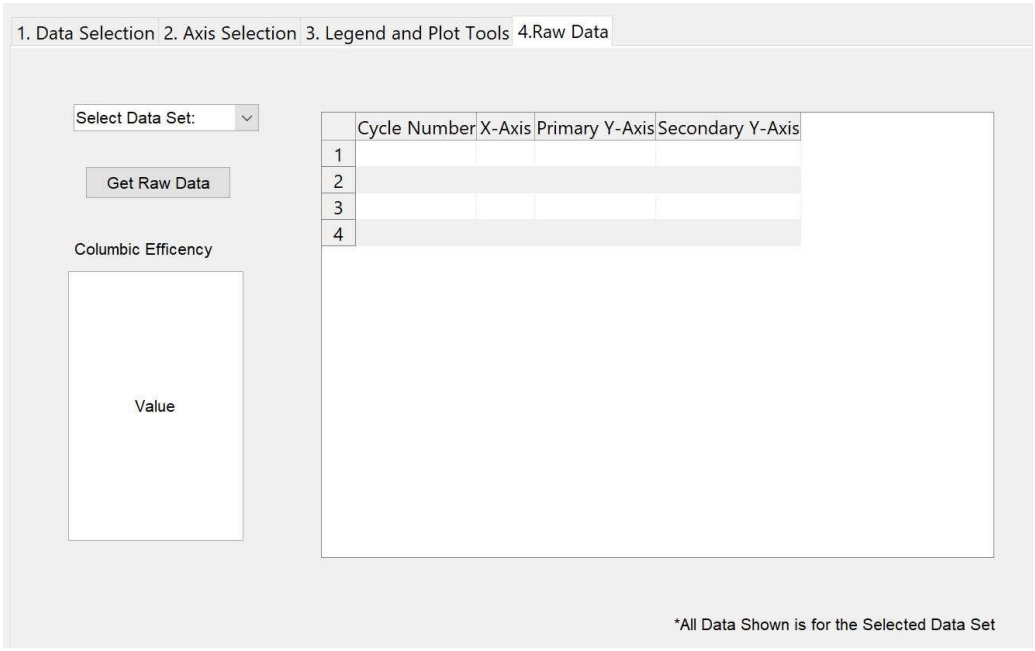

Figure 11: Raw Data Tab

5: Instructions

Initially, you should start in the first tab Data Selection:

1. You will first choose what data format they are working in: Excel or SQL.
2. In the Excel format, you will select the folder that you want to be your working directory.
3. You will then select the folder in which the excel files you are interested in are located.
4. Click “Select Folder”.
5. The data sets will appear in the list box, select the data sets that you want to analyze.
6. Click “Select Data Set(s)”.

Now you can navigate to the second tab, Axis Selection:

1. Select the Data Type for the X-Axis (i.e. Time)
2. You will make selections for sub-boxes of X-Axis Selection
    a. Choose whether you want charging, discharging, or both.
    b. For time you can choose the plot type (i.e. continuous)
    c. Choose if you want to put all cycles or just specific cycle in the Cycles to Plot panel
        i. Format for Cycle(s) box example: 1, 2, 3, 7-11, 21
3. Select the Y-Axis Data Type (i.e. Voltage)
4. Make selections on the Y-Axis Max and Y Axis Min.
    a. If you deselect the Auto you must put a valid number in for the range, otherwise you will get an error.
5. Repeat steps 3 and 4 for the Secondary Y-Axis if you so choose to add one.

Navigate to the Legend and Plot Tools Tab

1. If you select “Default All,” disregard all steps 2-XX otherwise continue.
2. If you deselect default Legend and Name
    a. In the legend setting box choose if you want the legend on or off
        i. Choose if you want the legend on the plot or off the plot
        ii. Choose which side of the plot you want the legend on. (i.e. Left, Right Top Bottom)
        iii. Move the slide bar to choose the location od the legend on that side. Far left is bottom and left and far right is top and right correlating to the side you choose to put the legend on
        iv. Choose font size of legend

b. Move on to Specific plot tools where you will make changes to one cycles worth of data to distinguish it from others.
    i. Choose both the data set you are working with and the cycle you are wanting to change
    ii. Choose the color you want the data to be
    iii. Make selections of marker type (i.e. circle, none)
    iv. Make selections of line type (i.e. solid, dashed, none)
    v. Change Legend name for the legend.
    vi. Click “Set Preferences”
    vii. Repeat this process for as many different cycles you want to change the settings for

3. If you deselect any other default settings, they will allow you to make changes in the General Plot Tools:
    a. Set the marker size if the value is not a number it will return an error
    b. Set the line width if the value is not a number it will return an error
    c. Set the size of the plot if these values are no numbers respectively it will return an error
4. Once all Legend and Plot Tools Have been set click “Plot”

Move to Raw Data Tab:

1. If you are unhappy with plot results you can repeat the process in that tab or you can export the data and get the raw values in this tab:
2. You will select from which data set you want to get the data in the dropdown menu
3. Click “Get Raw Data”
4. Raw Data will appear in the table according to the format described with headers for each column
5. You can select, and highlight using your mouse and Ctrl and Shift buttons to choose area and copy and paste into excel or another plot producing software to get plots that are paper ready.

Repeat the entire process as necessary to get all plots and data that you are interested in.

6:Graphs

There are four essential graphs to characterize a battery cell. Two of these graphs are shown over the cycle range: Columbic Efficiency vs Cycles and Discharge Specific Capacity vs Cycles. The other two are dQ/dV vs Voltage, and Discharge Specific Capacity vs. Voltage. These second two are used to show the characteristics of the battery and can be compared across cycles of the same battery or to different data sets. The coming sections will show an example of each, with a discussion about the process of graphing each one. All data shown is sample data provided by Dr. Kim [LNMO_PAA].

6.1: Columbic Efficiency vs. Cycles

Columbic efficiency is a measurement of the amount of energy produced from a battery over the amount of energy put in to the battery for that cycle. This graph is created by using the last and largest capacity value. Thus being, the total amount of energy inputted into the battery or given off by the battery. Once these values are found for each cycle the columbic efficiency is calculated by dividing the amount of energy given off by the battery by the amount of energy given to it for a given cycle. The closer this number is to 100% the better the performance of the battery is said to be. *Figure 12* shows a sample of this for all cycles in LNMO_PAA.

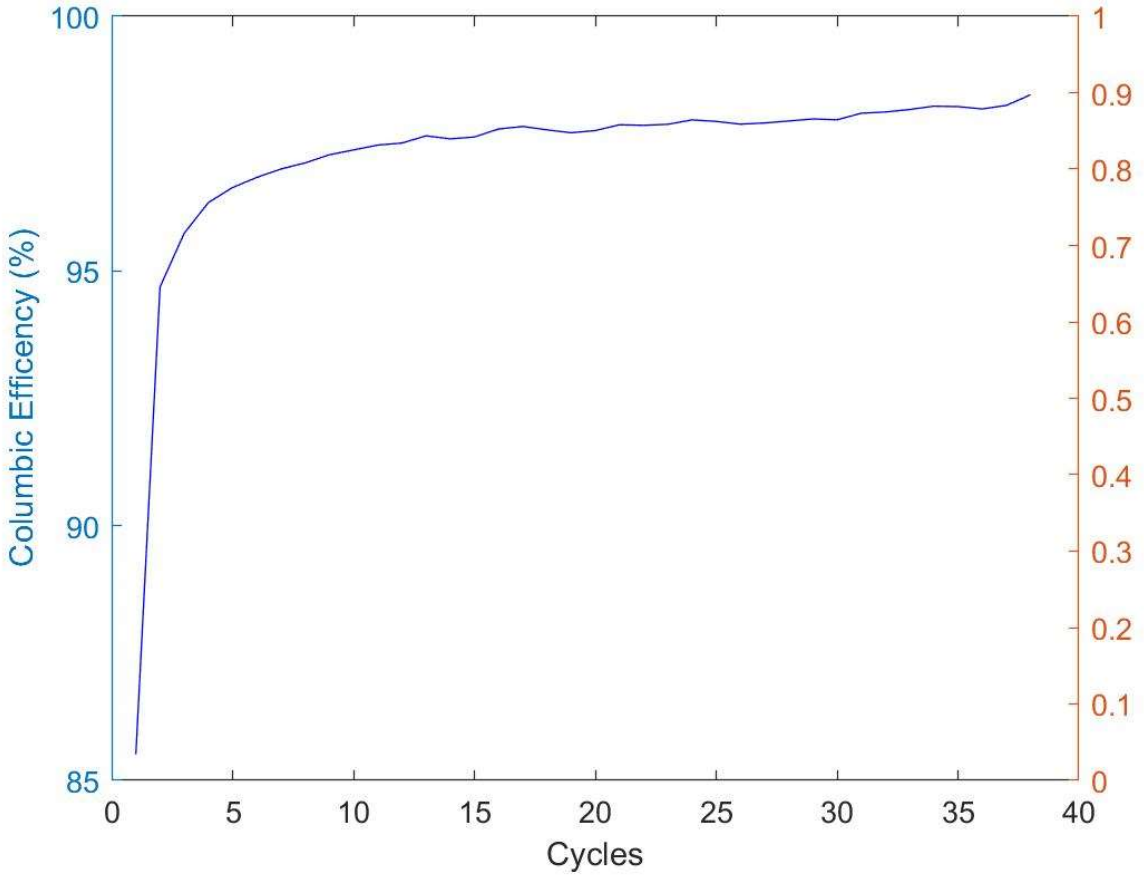

Figure 12: Sample Graph Columbic Efficiency vs. Cycles

6.2: Specific Capacity vs. Cycles

This graph is very similar to the first one although it measures the amount of energy that the cell takes in or gives off and divides it by the mass of the cell. For this graph the cell was given a mass of 15 mg and the top line represents the charge specific capacity and the bottom line represents the discharge specific capacity in mAh/g. This is what you would expect because if it was the other way around the battery would be producing more energy then what is put into it. An improvement to be made for this graph is add legends so the user can differentiate between the lines. *Figure 13* is also based off data from LNMMO_PAA.

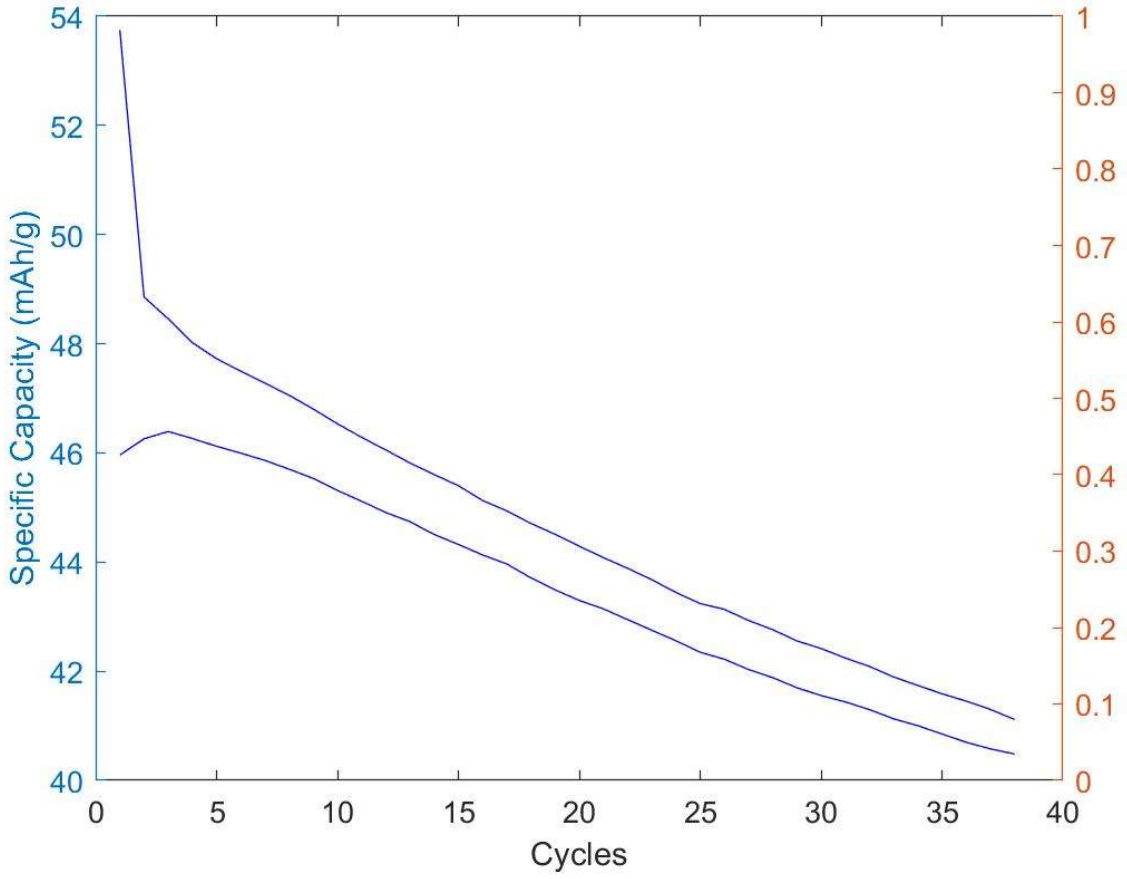

Figure 13: Sample Graph Specific Capacity vs. Cycles

6.3: dQ/dV vs. Voltage

The following plot shows the change in charge over change in voltage vs voltage. As you apply a current to a cell, the voltage will change. This plot is characteristic to each battery cell and should ideally remain relatively the same over the life of the battery. The top line is during charging, hence positive dQ/dV, and the bottom line is the discharging portion. *Figure 14* shows this for the 25th cycle in LNMO_PAA.

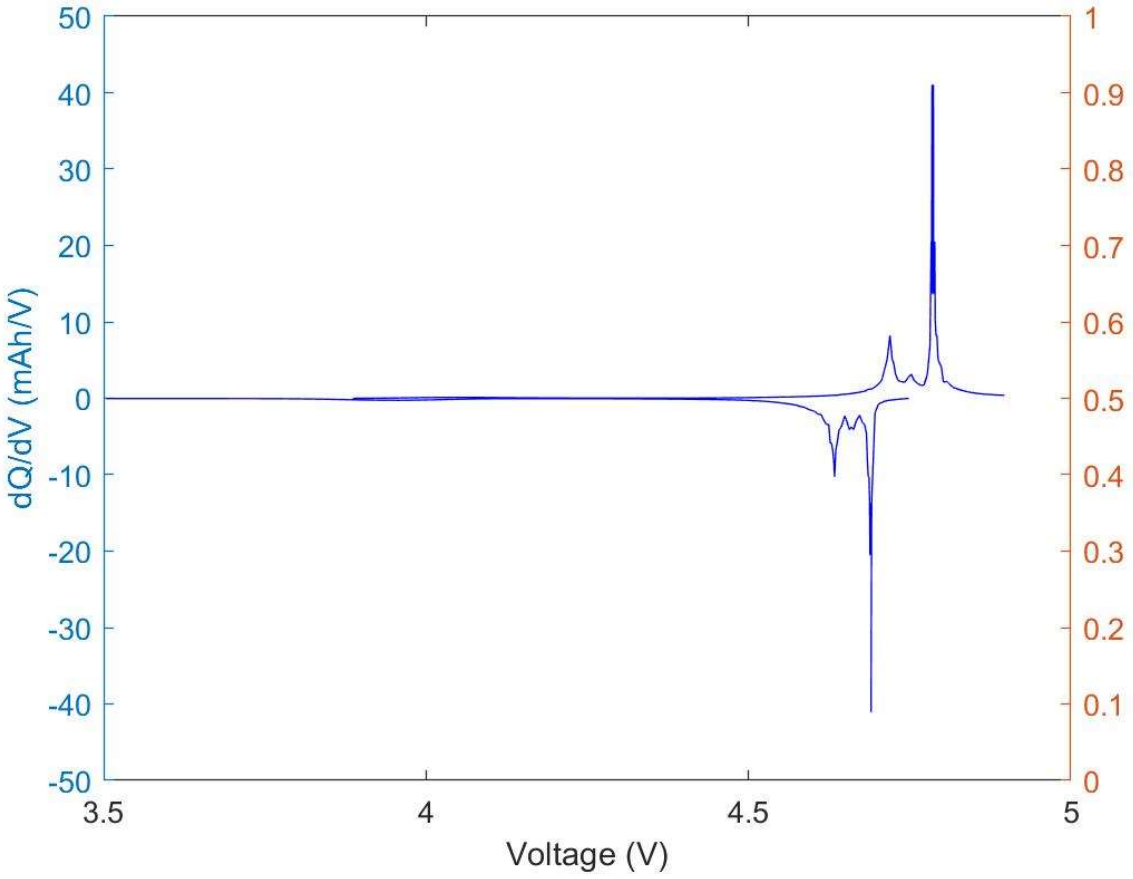

Figure 14: Sample Graph dQ/dV vs. Voltage for Cycle 25

6.4: Specific Capacity vs. Voltage

This plot is also a characteristic plot of each battery cell. It is the integral of the above plot. Since you are adding energy and removing energy from the battery at a constant rate it is very similar to the voltage over time graphs. The positive slope is the charging portion and the negative slope is the discharging portion. *Figure 15* shows an example for the 25th Cycle in LNMO_PAA. Once again to improve the readability of this graph a legend should be added.

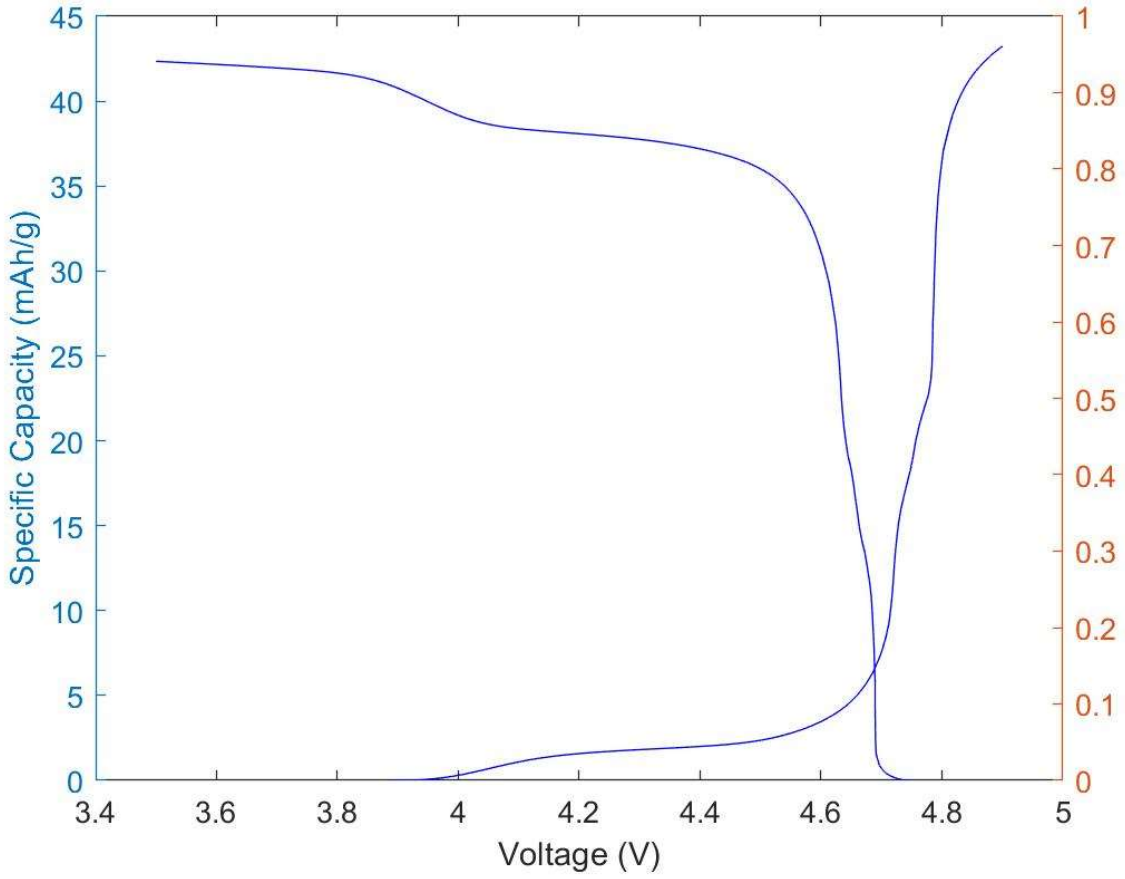

Figure 15: Sample Graph Specific Capacity vs Voltage for Cycle 25

6.5: Additional Plots

Below is one other basic plot that can be produced. The program has the option to produce many more graphs by choosing between all the different X-Axis and Y-Axis options. The plot below shows the use of the secondary axis by plotting both Current and Voltage vs. Time on one plot. This is very similar to the plot that it produced by Arbin immediately and was the first plot that this program could do. *Figure 16* only shows the 25th cycle as well, but you could choose to show more and to plot in the continuous domain, instead of the cyclic time domain.

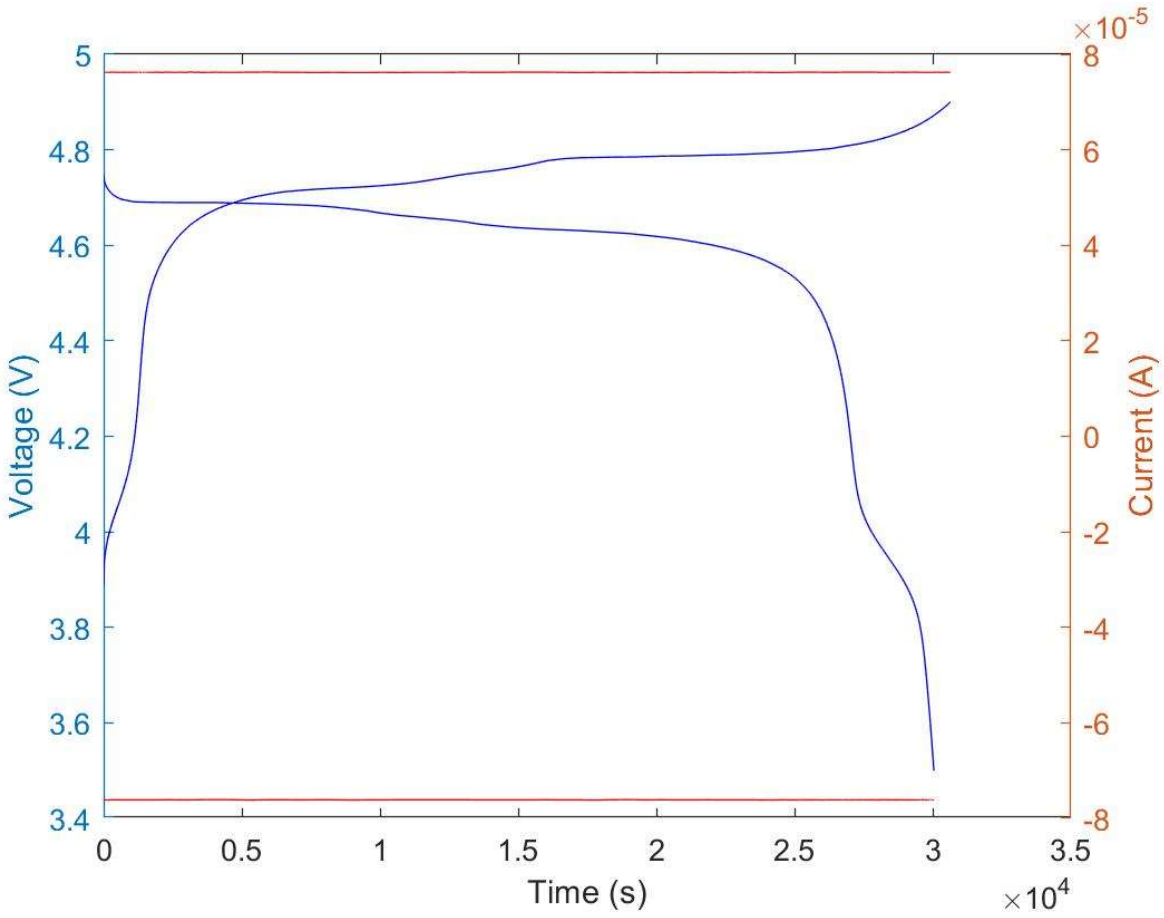

Figure 16: Sample Graph Voltage and Current vs. Time for Cycle 25

7: Programmer's Guide

To make it easier to add several functionalities to the program a programmer's guide has been created that explains each seperte component of the GUI and its programming background. The guid is broken up by panel and helps better explain how the programs interacts with MATLAB to make it easier for the development of the program.

7.1: Data Selection Tab

SQL Data Format: Radio button that when toggled on enables the functionality of SQL Data

Currently serves no purpose because SQL Data is not an option yet. Implement SQL Data panel and enable everything inside of it when in use.

Excel Data Format: Radio button that when toggled on enables the Excel Data panel

Allows the user to make all selections on working directory and data folder.

Browse WD Button: Button to choose where you want to put your working directory

Opens the computers directory and allows the user to select the location of the working directory. Functionality to save the working directory and load it the next time the program is open has not been added. Currently every time you open the program it starts both the working directory and the folder select to the C drive.

Working Directory: Editable textbox that contains the address for the working directory.

Set upon completion of the browsing of the working directory. Should be saved so that when the program is closed and then reopened it will remember where it was last.

Set WD: Button that allows the user to set the address that the folder select panel will start in

Must be clicked to transfer the Working directory to the select folder box. Should be consolidated to one textbox so this function could be removed.

Browse DF Button: Button to choose what folder the data is in

Opens the computers directory from the working directory and allows the user to select the location of the data folder.

Selected Folder: Editable textbox that contains the address for the selected folder.

Select Folder Button: Uses the address selected in the folder option to fill the Data Set list box.

Data Set: List box that is filled after the user clicks the Select Folder Button

Data Sets Button: Button to retrieve data from the specific battery cell tests selected in the Data Set list box.

Upon the clicking of this button the user will be asked to input in active mass for each test sample that they selected in the Data Set list box. Based of known format of excel file it will retrieve the data from the excel file that the user is interested in and sort the data into several different matrices. The excel data will be sorted between charge and discharge data and cycles. For each cycle the matrix will start a new column, for example in the ChVolt matrix the first column will include the charging data for the first cycle and so on for each cycle. A similar naming convention is done for all the variables charge and discharge. It is recommended that for speed purposes that some of this be moved into the data selection panel and only retrieve the raw data here and only sort the data that the user is interested in analyzing. In this portion of the code anytime there is no current being applied to the system the data is deleted because it is resting, this might not be advantageous for you, if so it can be removed. For each selected test a for loop is run and is stored in sequential cells at the end of the for loop. When a “Data” is added on to the end of a value it is the data for a single cycle and will only be a vector that is the length of the number of cycles. The user must wait for the wait bar to complete before the user can move on to the next process, consider removing this wait bar.

7.2: Axis Selection Tab

Axis Selection Tab

X Axis Data Type: Drop-Down Menu to choose the X Axis Data (e.g. time, voltage)

Upon selection it will enable or disable other components in the X-Axis Panel.

Cycles to Plot: Text box to input what cycles the user wants to plot

Charging Data: Button when selected selects the charging data can select both at the same time

Select X Axis: Uses all parameters selected in the panel to determine the X Axis data.

Once it determines this it will fill these handles for all different battery cycle tests. If invalid selection is made an error will be displayed. Addition of data types would be in the XAVal area and another else if would be added to each section. Currently Specific Capacity is calculated in this section but everything else is organized in the Select Data Sets callback function.

Discharging Data: Button when selected selects the discharging data

Max Range P: Set the max range of the primary Y Axis in textbox

No functionality yet.

Auto P1: Set the max range to Auto so the plot decides the highest value for the primary y axis

Disables the Max Range P textbox

Min Range P: Set the min range of the primary Y Axis in textbox

No functionality yet

Auto P2: Set the min range to Auto so the plot decides the lowest value for the primary y axis

Disables the Min Range P textbox

Secondary Axis Add: Checkbox that allows the user to add another Y-Axis

Upon clicking it will either enable or disable all selections for the secondary Y Axis. All function for the secondary Y Axis are described below

Y Axis Data Type 2: Drop-Down Menu to choose the secondary Y Axis Data (e.g. capacity)

Max Range S: Set the max range of the secondary Y Axis in textbox

No functionality yet.

Auto S1: Set the max range to Auto so the plot decides the highest value for the secondary y axis

Disables the Max Range S textbox

Min Range S: Set the min range of the secondary Y Axis in textbox

No functionality yet

Auto S2: Set the min range to Auto so the plot decides the lowest value for the secondary y axis

Disables the Min Range S textbox

Y Axis Data Type 1: Drop-Down Menu to choose the primary Y Axis Data (e.g. capacity)

Select Y Axis: Uses all parameters selected in the panel to determine the Y Axis data.

Loads entire set of data and does not sort out which cycles are being plotted this is done only on the X Axis and then when the plot button is selected it determines which Y Axis data is necessary depending on which columns are completely empty when plotting. This is also when the data is sorted for raw data tab.

Select SY Axis: Uses all parameters selected in the panel to determine the SY Axis data

Like the Select Y Axis button it selects the entire data set and does the raw data at the plot selection.

7.3: Legend and Plot Tools Tab

Legend and Plot Tools Tab

Default All Plot: Checkbox in defaults panel that when selected disables all functions in the Legend and Plot Tools panel

Currently the only option for plotting the data below will describe the idea for each function in this panel and will be updated as components gain functionality.

Default Legend Plot: Checkbox that in defaults panel that when deselected enables Legend options.

Enables all components in the Legend Settings panel and everything but the Legend Name textbox in the specific plot tools panel.

Default Line Plot: Checkbox that in defaults panel that when deselected enables line width option.

Default Markers Plot: Checkbox that is in defaults panel that when deselected enables marker size option.

Default Name Plot: Checkbox that is in defaults panel that when deselected enables legend name option.

Default Size Plot: Checkbox that is in defaults panel that when deselected enables plot size options to change the size of the plot.

Line Width Plot: editable text box in General Plot Tools panel that allows the user to input a number for the line width for all the lines in the plot.

Currently has no functionality. Should be called out when the user selects the plot button.

Marker Size Plot: editable text box in General Plot Tools panel that allows the user to input a number for the marker size for all the markers in the plot.

Plot Height: editable text box in General Plot Tools panel that allows the user to input a number for the height of the plot they want to produce in centimeters.

Legend Location: Drop-down menu that allows the user to select which side of the plot they want the legend on.

Currently has no functionality. Should be called out when the user selects the plot button.

Position: Slide bar that allows the user to choose where on the side they want the plot

Currently has no functionality. The left side of the bar corresponds to the bottom or left depending on which side they select and should be a percentage of the max position. Should be called out when the user selects the plot button.

Legend Font Size: editable text box in Legend Settings panel that allows the user to input a number for the font size of the legend.

Off Graph Check: Check box in Legend Settings panel that allows the user to place the legend box off the graph.

Currently has no functionality. This option will make the plot smaller because it will take up space for just the legend. Should be called out when the user selects the plot button.

Legend On Check: Check box in Legend Settings panel that allows the user to toggle on or off a legend.

Currently has no functionality. Should be called out when the user selects the plot button.

Select Data Plot: Drop-down menu in the Specific Plot Tools panel that allows the user to select the battery test data they are interested in changing.

When selected the Select Cycle Plot drop-down menu is populated with the proper cycles based on selections in the Axis Selection Tab. Once the user selects the test they Select Cycle Plot drop-down menu the user can then select from that drop-down menu.

Select Cycle Plot: Drop-down menu in the Specific Plot Tools panel that allows the user to select the cycle from the specific battery test they are interested in changing.

Currently has no functionality. When the user selects cycle each box should be auto populated based off the default settings. The user can then change these default settings.

Color Select Plot: Drop-down menu in the Specific Plot Tools panel that allows the user to select the color for the plot of that cycle.

Currently has no functionality. Could be replaced by a color wheel or something similar to give several options for the color of the plot. Saved when the user selects the “Set Preferences” button.

Marker Type Plot: Drop-down menu in the Specific Plot Tools panel that allows the user to select the marker for the plot of that cycle.

Currently has no functionality. Saved when the user selects the “Set Preferences” button.

Line Type Plot: Drop-down menu in the Specific Plot Tools panel that allows the user to select the line type for the plot of that cycle.

Currently has no functionality. Saved when the user selects the “Set Preferences” button.

Set Preferences: Button to allow the user to set the settings for the given cycle number and battery test.

Currently has no functionality. When implemented it should reset all prior functions so the user knows its set. When the user reloads a specific cycle all the information should show up as it is now not s default settings.

Legend Name: Editable text box that allows the user to change the name of the specific cycle selected as it will be displayed on the legend.

Currently has no functionality. The following should be default filled with the test name from the Data Sets box and then the cycle number given. This will not distinguish between charging and discharging data. The user will have to know what data they selected.

Plot Button: use the all settings determined before to plot the data and add a legend accordingly. XAxisA is for cycle numbers. XAxis1 is for charging data and XAxis2 is for discharging data. YAxis1 can either be cycle data or charging data correspondingly. YAxis2 is for discharging data. Similarly adding the S is for the secondary Y Axis. Depending on which data sets are empty for a given test it determines what you must plot. If a given column is completely empty it will skip it and not plot it. Should add functionality to identify each plot and be able to make changes to the plot after to make the set preferences. The other option is to use the set preferences while plotting simultaneously. It then saves the data into raw data handles. With similar naming conventions. These data sets are organized to know exactly what data was plotted. It then sets the X Label and Y Label accordingly dependent on the selections.

7.4: Raw Data Tab

Raw Data Set: Drop down menu user selects the battery cell test that they want the data from

Columbic Efficiency: List box that displays the columbic efficiency for each cycle upon the clicking of the Raw Data button

This currently has no functionality, but the values have already been calculated it is matter of calling the right variable after the button is clicked

Raw Data: Uses the saved handles in the plot button call back for raw data to populate the Raw Data table. The format is described in the paper with the X Axis for the first cycle being plotted for charge then Y Axis and secondary Y Axis. The discharge for the first cycle is then plotted in the next columns. Then the following cycles that are placed in the table in the same format.

8: Conclusion

The goal of this research was to automate the entire process of data analysis for Li-ion battery data. This automation starts with formatting the raw data and graphing the raw data in a usable and understandable way. Not only was it favorable to automate this process but also make it seamless and give the user this ability with no coding knowledge. This automation was completed by using MATLAB GUI (graphic user interface). This program allows the user to select several different options and graph several different plots by following the instructions described above. This can all be done with a few clicks so if the user has ability to perform basic functions of a computer the user can utilize this program. Ultimately all basic goals of this project were met, by automating data analysis in a way that makes the it feasible for all.

8.1: Contributions

The biggest contribution that this software provides is a way for Ohio State battery researchers to complete data analysis quickly and efficiently. It is estimated that to produce the essential graphs in excel would take several hours just to choose all the data for one cell. Much more time would be taken if trying to compare across several cells. All of this can be done in around 10 minutes with the program that was designed in this research. This is huge for a researcher, as they can spend much more times in other aspects of research by getting results faster. Eventually this will be shared with several universities that use Arbin cyclers to reap the benefits of the program across the community. This will also allow others to make improvements and share them.

8.2: Additional Work

This type of philosophy in data analysis can be applied to almost all fields of research and save on how many times you do repetitive processes. If you can create a program to automate several different data analysis techniques, you can save time on monotonous tasks. This has

become a very valuable part of research because at universities time may not be money, but is valuable in producing worthy research and anywhere time could be saved can improve the research. Every group at Ohio State and universities across the nation should automate data analysis to the best of its abilities.

8.3: Future Work

There are several components to this program that could be improved and added. Improvements should be in the time aspect and creating the most optimized process/code. Additions will include SQL ability, and legend functions.

Many aspects of the code are not optimized to save time. For instance, the process of sorting the data could be moved to the Axis selection tab to improve the time it takes to read the data in. This would remove any unessential lines. For instance, if you never use the “time” data in your plots and it still sorts it you can save time by not sorting this data unless the user asks for it. This is one example of this and the solution done for the software is not optimized, but it works. There are many other components of the program that need to be optimized so the user is not waiting excessive amounts of time.

Of the additional pieces to be added, the most essential piece is the SQL database organization. As previously stated, the computer that houses the SQL database is not hooked up to the network and is not advantageous to pursue this. When it is hooked up it will be worth it to pursue this and be able to get data straight from the database without having to convert it to Excel first. One of the hardest parts of this is figuring out where the data for each test is stored and how to query the data and sort it in MATLAB once it is queried correctly. Another piece to be added is the legend plots, now that you can plot everything, adding more plot control to

produce presentation ready plots is the final outcome. If it can have so many options that they do not have to export the data and reconfigure it, you can save the researchers even more time.

8.4 Summary

The basic needs of the program are complete, and much of the improvements that can be made to it are small cosmetic pieces. These will take little time to add, but the users will benefit greatly from them. The project was fulfilling, when a final product was produced. The middle stages where the interface was created but no functionality was added to it was hard to get through. Errors would continue to populate and discourage further development. Since much of these errors have been mitigated, the new programmers can take the time to study it and add to it as they see fit. Finally, if someone wanted to help create a better understanding for this program, it should be split up into different functions to make it easier to navigate the code. Right now the programmers guide breaks down each separate component to the program and should be added on to as necessary. Most improvements to be made will take little time because a good basis has been created. Each component should be added quickly.

Appendix A Copy of Code:

To request a copy of the code contact me at ferrato.3@osu.edu Dr. Kim at kim.6776@osu.edu or Cody O'Meara omeara.31@osu.edu

Appendix B Raw Code:

```
function varargout = BatteryDataAnalysisGUI2(varargin)
% BATTERYDATAANALYSISGUI2 MATLAB code for BatteryDataAnalysisGUI2.fig
%      BATTERYDATAANALYSISGUI2, by itself, creates a new
BATTERYDATAANALYSISGUI2 or raises the existing
%      singleton*.
%
%      H = BATTERYDATAANALYSISGUI2 returns the handle to a new
BATTERYDATAANALYSISGUI2 or the handle to
%      the existing singleton*.
%
%      BATTERYDATAANALYSISGUI2('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in BATTERYDATAANALYSISGUI2.M with the given input
arguments.
%
%      BATTERYDATAANALYSISGUI2('Property','Value',...) creates a new
BATTERYDATAANALYSISGUI2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before BatteryDataAnalysisGUI2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to BatteryDataAnalysisGUI2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help BatteryDataAnalysisGUI2

% Last Modified by GUIDE v2.5 26-Mar-2018 21:51:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @BatteryDataAnalysisGUI2_OpeningFcn, ...
                   'gui_OutputFcn',  @BatteryDataAnalysisGUI2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT


% --- Executes just before BatteryDataAnalysisGUI2 is made visible.
function BatteryDataAnalysisGUI2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to BatteryDataAnalysisGUI2 (see VARARGIN)

% Choose default command line output for BatteryDataAnalysisGUI2
handles.output = hObject;

%Intilise Tab Manager
handles.tabManager = TabManager( hObject );

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes BatteryDataAnalysisGUI2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = BatteryDataAnalysisGUI2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in XAxisDataType.
function XAxisDataType_Callback(hObject, eventdata, handles)
% hObject    handle to XAxisDataType (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns XAxisDataType contents as cell array
%        contents{get(hObject,'Value')} returns selected item from XAxisDataType
XAVal=get(handles.XAxisDataType,'Value')
if XAVal==2
    set(findall(handles.CyclesToPlotPanel, '-property', 'enable'), 'enable', 'on')
    set(findall(handles.PlotTypeButton, '-property', 'enable'), 'enable', 'on')
    set(handles.ChargingData,'Enable','on')
    set(handles.DischargingData,'Enable','on')
elseif XAVal==3
    set(findall(handles.CyclesToPlotPanel, '-property', 'enable'), 'enable', 'on')
```

```
    set(findall(handles.PlotTypeButton, '-property', 'enable'), 'enable', 'off')
    set(handles.ChargingData,'Enable','on')
    set(handles.DischargingData,'Enable','on')
elseif XAVal==4
    set(findall(handles.CyclesToPlotPanel, '-property', 'enable'), 'enable', 'off')
    set(findall(handles.PlotTypeButton, '-property', 'enable'), 'enable', 'off')
    set(handles.ChargingData,'Enable','off')
    set(handles.DischargingData,'Enable','off')
    set(handles.AllButton,'Value',1)
    set(handles.CyclesButton,'Value',0)
    set(handles.ChargingData,'Value',0)
    set(handles.DischargingData,'Value',0)
elseif XAVal==5
    set(findall(handles.CyclesToPlotPanel, '-property', 'enable'), 'enable', 'on')
    set(findall(handles.PlotTypeButton, '-property', 'enable'), 'enable', 'off')
    set(handles.ChargingData,'Enable','on')
    set(handles.DischargingData,'Enable','on')
elseif XAVal==6
    set(findall(handles.CyclesToPlotPanel, '-property', 'enable'), 'enable', 'on')
    set(findall(handles.PlotTypeButton, '-property', 'enable'), 'enable', 'off')
    set(handles.ChargingData,'Enable','on')
    set(handles.DischargingData,'Enable','on')
else
    set(findall(handles.CyclesToPlotPanel, '-property', 'enable'), 'enable', 'off')
    set(findall(handles.PlotTypeButton, '-property', 'enable'), 'enable', 'off')
    set(handles.ChargingData,'Enable','off')
    set(handles.DischargingData,'Enable','off')
    set(handles.AllButton,'Value',1)
    set(handles.CyclesButton,'Value',0)
    set(handles.ChargingData,'Value',0)
    set(handles.DischargingData,'Value',0)
end


% --- Executes during object creation, after setting all properties.
function XAxisDataType_CreateFcn(hObject, eventdata, handles)
% hObject    handle to XAxisDataType (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function CyclesToPlot_Callback(hObject, eventdata, handles)
% hObject    handle to CyclesToPlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of CyclesToPlot as text
%        str2double(get(hObject,'String')) returns contents of CyclesToPlot as a double


% --- Executes during object creation, after setting all properties.
function CyclesToPlot_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CyclesToPlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in YAxisDataType2.
function YAxisDataType2_Callback(hObject, eventdata, handles)
% hObject    handle to YAxisDataType2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns YAxisDataType2 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from YAxisDataType2


% --- Executes during object creation, after setting all properties.
function YAxisDataType2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to YAxisDataType2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function MaxRangeS_Callback(hObject, eventdata, handles)
% hObject    handle to MaxRangeS (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MaxRangeS as text
%        str2double(get(hObject,'String')) returns contents of MaxRangeS as a double
```

```
% --- Executes during object creation, after setting all properties.
function MaxRangeS_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MaxRangeS (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in AutoS1.
function AutoS1_Callback(hObject, eventdata, handles)
% hObject    handle to AutoS1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of AutoS1
Auto=get(handles.AutoS1,'Value');
if Auto==1
    set(handles.MaxRangeS,'Enable','off')
else
    set(handles.MaxRangeS,'Enable','on')
end


function MinRangeS_Callback(hObject, eventdata, handles)
% hObject    handle to MinRangeS (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MinRangeS as text
%        str2double(get(hObject,'String')) returns contents of MinRangeS as a double


% --- Executes during object creation, after setting all properties.
function MinRangeS_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MinRangeS (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in AutoS2.
```

```
function AutoS2_Callback(hObject, eventdata, handles)
% hObject    handle to AutoS2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of AutoS2
Auto=get(handles.AutoS2,'Value');
if Auto==1
   set(handles.MinRangeS,'Enable','off')
else
   set(handles.MinRangeS,'Enable','on')
end

% --- Executes on selection change in YAxisDataType1.
function YAxisDataType1_Callback(hObject, eventdata, handles)
% hObject    handle to YAxisDataType1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns YAxisDataType1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from YAxisDataType1


% --- Executes during object creation, after setting all properties.
function YAxisDataType1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to YAxisDataType1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function MaxRangeP_Callback(hObject, eventdata, handles)
% hObject    handle to MaxRangeP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MaxRangeP as text
%        str2double(get(hObject,'String')) returns contents of MaxRangeP as a double


% --- Executes during object creation, after setting all properties.
function MaxRangeP_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MaxRangeP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in AutoP1.
function AutoP1_Callback(hObject, eventdata, handles)
% hObject    handle to AutoP1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of AutoP1
Auto=get(handles.AutoP1,'Value');
if Auto==1
    set(handles.MaxRangeP,'Enable','off')
else
    set(handles.MaxRangeP,'Enable','on')
end


function MinRangeP_Callback(hObject, eventdata, handles)
% hObject    handle to MinRangeP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MinRangeP as text
%        str2double(get(hObject,'String')) returns contents of MinRangeP as a double


% --- Executes during object creation, after setting all properties.
function MinRangeP_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MinRangeP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in BrowseWDButton.
function BrowseWDButton_Callback(hObject, eventdata, handles)
% hObject    handle to BrowseWDButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
WD=get(handles.WorkingDirectory,'String');
```

```
source_dir = uigetdir(WD);
set(handles.WorkingDirectory,'String',(source_dir))


function WorkingDirectory_Callback(hObject, eventdata, handles)
% hObject    handle to WorkingDirectory (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of WorkingDirectory as text
%        str2double(get(hObject,'String')) returns contents of WorkingDirectory as a double


% --- Executes during object creation, after setting all properties.
function WorkingDirectory_CreateFcn(hObject, eventdata, handles)
% hObject    handle to WorkingDirectory (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in SetWD.
function SetWD_Callback(hObject, eventdata, handles)
% hObject    handle to SetWD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
WD=get(handles.WorkingDirectory,'String');
set(handles.SelectedFolder,'String',WD)


% --- Executes on selection change in RawDataSet.
function RawDataSet_Callback(hObject, eventdata, handles)
% hObject    handle to RawDataSet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns RawDataSet contents as cell array
%        contents{get(hObject,'Value')} returns selected item from RawDataSet


% --- Executes during object creation, after setting all properties.
function RawDataSet_CreateFcn(hObject, eventdata, handles)
% hObject    handle to RawDataSet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ColumbicEfficency_Callback(hObject, eventdata, handles)
% hObject    handle to ColumbicEfficency (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ColumbicEfficency as text
%        str2double(get(hObject,'String')) returns contents of ColumbicEfficency as a double


% --- Executes during object creation, after setting all properties.
function ColumbicEfficency_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ColumbicEfficency (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in SecondaryAxisAdd.
function SecondaryAxisAdd_Callback(hObject, eventdata, handles)
% hObject    handle to SecondaryAxisAdd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of SecondaryAxisAdd
Val=get(handles.SecondaryAxisAdd,'Value');
if Val==1
    set(findall(handles.SecondaryAxis, '-property', 'enable'), 'enable', 'on')
    set(handles.SelectSYAxis,'Enable','on')
else
    set(findall(handles.SecondaryAxis, '-property', 'enable'), 'enable', 'off')
    set(handles.SelectSYAxis,'Enable','off')
end

% --- Executes on selection change in DataSet.
function DataSet_Callback(hObject, eventdata, handles)
% hObject    handle to DataSet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: contents = cellstr(get(hObject,'String')) returns DataSet contents as cell array
%        contents{get(hObject,'Value')} returns selected item from DataSet


% --- Executes during object creation, after setting all properties.
function DataSet_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DataSet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in DataSetsButton.
function DataSetsButton_Callback(hObject, eventdata, handles)
% hObject    handle to DataSetsButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Retrieve handle variables
source_files = handles.source_files;
PMass = handles.PMass;
Data = handles.data;
source_dir = handles.source_dir;

% Set Variables (the columns in the Excel sheet where the corresponding data is)
currentCol = 7;
voltCol = 8;
chargeCol = 9;
dischargeCol = 10;
timeCol=2;

% Get the selected filename(s) from listbox
selectedValue = get(handles.DataSet, 'Value');
list = get(handles.DataSet, 'String');

% Preallocate for speed
prompt = cell(length(selectedValue), 1);
defaultans = cell(length(selectedValue), 1);

% User enters active mass for selected filename
for ii = 1:length(selectedValue)
    dataName = list{selectedValue(ii)};
    prompt{ii} = ['Enter ', dataName, ' active mass [mg]:'];
    if isnan(PMass(selectedValue(ii)))
        defaultans{ii} = {'0'};
    else
```

```
        defaultans{ii} = {num2str(PMass(selectedValue(ii)) * 1000)};
    end
end
dlg_title = 'Enter Mass';
num_lines = 1;
answer = inputdlg(prompt, dlg_title, num_lines, [defaultans{:}]);

for ii = 1:length(selectedValue)
    % Store the selected Excel File(s)
    Data{selectedValue(ii)} = xlsread(fullfile(source_dir, source_files(selectedValue(ii)).name),2);
    PMass(selectedValue(ii)) = str2double(answer{ii}) * 10^-3;
end
for zz = 1:length(selectedValue)
    data = Data{selectedValue(zz)};
    %Delete Resting Data
    data(mod(data(:,currentCol), 0) == 0, :) = [];

    % Calculate number of charge and discharge cycles
    tot = 2 * length(find(diff(data(:,currentCol)) > data(1,currentCol))) + 2;

    % Split Charge & Discharge Data
    ChCap = nan(length(data), tot);
    ChVolt = nan(length(data), tot);
    ChTime = nan(length(data), tot);
    DisCap = nan(length(data), tot);
    DisVolt = nan(length(data), tot);
    DisTime = nan(length(data), tot);
    ChCycTime = nan(length(data), tot);
    DisCycTime = nan(length(data), tot);
    ChCurrent = nan(length(data), tot);
    DisCurrent = nan(length(data), tot);

    % Get Charging Info
    ii = 1;
    kk = 1;
    hwb = waitbar(0, 'Retrieving Data...', 'Name', 'Importing');
    for jj = 1:length(data)
        if data(jj, currentCol) > 0  % If the current is positive, it is charging
            ChCap(kk,ii) = data(jj,chargeCol).*1000;
            ChVolt(kk,ii) = data(jj,voltCol);
            ChTime(kk,ii) = data(jj,timeCol);
            ChCurrent(kk,ii) = data(jj,currentCol);
            kk = kk + 1;
            if jj < length(data) && data(jj,currentCol) * data(jj+1,currentCol) < 0
                ii = ii+1;
                kk = 1;
            end
        end
        waitbar(jj/length(data), hwb);
    end
    close(hwb)
```

```
% Get Discharging Info
ii = 1;
kk = 1;
for jj = 1:length(data)
    if data(jj,currentCol) < 0  % If the current is negative, it is discharging
        DisCap(kk,ii) = data(jj,dischargeCol).*1000;
        DisVolt(kk,ii) = data(jj,voltCol);
        DisTime(kk,ii) = data(jj,timeCol);
        DisCurrent(kk,ii) = data(jj,currentCol);
        kk = kk + 1;
        if jj < length(data) && data(jj,currentCol) * data(jj+1,currentCol) < 0
            ii = ii+1;
            kk = 1;
        end
    end
end

% Create cycle times
ChCycTime=ChTime-ChTime(1,:);
DisCycTime=DisTime-DisTime(1,:);

% If there's no data in a column, delete the column
ChCap = ChCap(:,sum(~isnan(ChCap))~=0);
ChVolt = ChVolt(:,sum(~isnan(ChVolt))~=0);
ChTime = ChTime(:,sum(~isnan(ChTime))~=0);
DisCap = DisCap(:,sum(~isnan(DisCap))~=0);
DisVolt = DisVolt(:,sum(~isnan(DisVolt))~=0);
DisTime = DisTime(:,sum(~isnan(DisTime))~=0);
ChCycTime = ChCycTime(:,sum(~isnan(ChCycTime))~=0);
DisCycTime = DisCycTime(:,sum(~isnan(DisCycTime))~=0);
ChCurrent = ChCurrent(:,sum(~isnan(ChCurrent))~=0);
DisCurrent = DisCurrent(:,sum(~isnan(DisCurrent))~=0);

% Determine number of full cycles for preallocation
[chR, chCol] = size(ChCap);
[disR, disCol] = size(DisCap);
rows = max(chR, disR);
cycles = min(chCol, disCol);

% Preallocate for speed
[RV CV]=size(ChVolt);
[RD CD]=size(DisVolt);
chdQdV = nan(RV, CV);
disdQdV = nan(RD, CD);

% If capacity wasn't reset, this will reset it
for aa = 1:chCol
    if min(ChCap(:,aa)) > 0
        ChCap(:,aa) = ChCap(:,aa) - min(ChCap(:,aa));
    end
```

```
end
for aa = 1:disCol
    if min(DisCap(:,aa)) > 0
        DisCap(:,aa) = DisCap(:,aa) - min(DisCap(:,aa));
    end
end

% Specify the specific capacities for charging and discharging
chSpecCap = ChCap./PMass(selectedValue(zz));
disSpecCap = DisCap./PMass(selectedValue(zz));

% Store Data in cells
chCap{zz} = ChCap;
chVolt{zz} = ChVolt;
chTime{zz} = ChTime;
disCap{zz} = DisCap;
disVolt{zz} = DisVolt;
disTime{zz} = DisTime;
chCycTime{zz} = ChCycTime;
disCycTime{zz} = DisCycTime;

ChSpecCap{zz} = chSpecCap;
DisSpecCap{zz} = disSpecCap;
ChdQdV{zz} = chdQdV;
DisdQdV{zz} = disdQdV;
chCurrent{zz} = ChCurrent;
disCurrent{zz} = DisCurrent;

if cycles >= 1   % If there's at least 1 full cycle

    % Preallocate for speed
    chCapData = nan(cycles, 1);
    disCapData = nan(cycles, 1);
    coulEff = nan(cycles, 1);
    cycNum = nan(cycles, 1);
    legendStr = strings(cycles, 1);
    colorVals = zeros(cycles, 3);
    colorVals(:,1) = .733;              % Make default color 'Scarlet'
    if length(selectedValue) == 1
        cycStr2 = strings(cycles+1, 1);
    else
        cycStr2 = strings(length(selectedValue), 1);
    end

    % Get the maximum capacity for each cycle
    for ll = 1:cycles
        chSpecCapData(ll)=max(ChCap(:,ll))./PMass(selectedValue(zz));
        disSpecCapData(ll)=max(DisCap(:,ll))./PMass(selectedValue(zz));
        chCapData(ll) = max(ChCap(:,ll));
        disCapData(ll) = max(DisCap(:,ll));
        cycNum(ll,1) = ll;
```

```
        end

        % Calculate Coulombic Efficiency for each cycle
        for zzz = 1:cycles
            coulEff(zzz,1) = (disCapData(zzz,1)/chCapData(zzz,1)).*100;
        end
    % If there isn't a full cycle
    else
%         set(handles.ceEdit,'String', 'There is not a full cycle for this data')
        coulEff = nan;
        disCapData = nan;
        cycNum = nan;
        legendStr = strings(1,1);
        colorVals = zeros(1, 3);
        colorVals(:,1) = .733;
%         set(handles.cycleMenu, 'String', num2str(1));
%         set(handles.cycleEdit, 'String', ' ');
    end
    ChCapData{zz} = chCapData;
    DisCapData{zz} = disCapData;
    ChSpecCapData{zz} = chSpecCapData;
    DisSpecCapData{zz} = disSpecCapData;
    CycNum{zz} = cycNum;
    CoulEff{zz} = coulEff;
    ColorVals{zz} = colorVals;
    Cycles{zz} = cycles;

end

% Store Variables
handles.coulEff = CoulEff;
handles.cycles = cell2mat(Cycles);
handles.dataName = dataName;
handles.cycNum = CycNum;
handles.chdQdV = ChdQdV;
handles.disdQdV = DisdQdV;
handles.chCapData =ChCapData;
handles.disCapData = DisCapData;
handles.PMass = PMass;
handles.selectedValue = selectedValue;

handles.ChCap = chCap;
handles.ChVolt = chVolt;
handles.ChTime = chTime;
handles.ChCycTime =chCycTime;
handles.DisCap = disCap;
handles.DisVolt = disVolt;
handles.DisTime = disTime;
handles.DisCycTime = disCycTime;
handles.chSpecCap = ChSpecCap;
handles.disSpecCap =DisSpecCap;
```

```
handles.chSpecCapData= ChSpecCapData;
handles.disSpecCapData= DisSpecCapData;
handles.ChCurrent= chCurrent;
handles.DisCurrent= disCurrent;
handles.legendStr = legendStr;
handles.colorVals = ColorVals;
handles.chCapData = ChCapData;
handles.disCapData =DisCapData;

handles.list = list;

guidata(hObject,handles)

% --- Executes on button press in DefaultAllPlot.
function DefaultAllPlot_Callback(hObject, eventdata, handles)
% hObject    handle to DefaultAllPlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of DefaultAllPlot
DAVal=get(handles.DefaultAllPlot,'Value');
if DAVal==1
   set(handles.DefaultLegendPlot,'Enable','off')
   set(handles.DefaultLegendPlot,'Value',1)
   set(handles.DefaultLine,'Enable','off')
   set(handles.DefaultLine,'Value',1)
   set(handles.DefaultMarkersPlot,'Enable','off')
   set(handles.DefaultMarkersPlot,'Value',1)
   set(handles.DefaultNamePlot,'Enable','off')
   set(handles.DefaultNamePlot,'Value',1)
   set(handles.DefaultSizePlot,'Enable','off')
   set(handles.DefaultSizePlot,'Value',1)
   set(findall(handles.LocationPanel, '-property', 'enable'), 'enable', 'off')
   set(findall(handles.PlotToolsPanel1, '-property', 'enable'), 'enable', 'off')
   set(findall(handles.PlotToolsPanel2, '-property', 'enable'), 'enable', 'off')
else
   set(handles.DefaultLegendPlot,'Enable','on')
   set(handles.DefaultLegendPlot,'Value',1)
   set(handles.DefaultLine,'Enable','on')
   set(handles.DefaultLine,'Value',1)
   set(handles.DefaultMarkersPlot,'Enable','on')
   set(handles.DefaultMarkersPlot,'Value',1)
   set(handles.DefaultNamePlot,'Enable','on')
   set(handles.DefaultNamePlot,'Value',1)
   set(handles.DefaultNamePlot,'Enable','on')
   set(handles.DefaultNamePlot,'Value',1)
   set(handles.DefaultSizePlot,'Enable','on')
   set(handles.DefaultSizePlot,'Value',1)
end
% --- Executes on button press in DefaultLegendPlot.
function DefaultLegendPlot_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to DefaultLegendPlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of DefaultLegendPlot
DLVal=get(handles.DefaultLegendPlot,'Value');
if DLVal==1
   set(findall(handles.LocationPanel, '-property', 'enable'), 'enable', 'off')
   set(findall(handles.PlotToolsPanel1, '-property', 'enable'), 'enable', 'off')
   set(handles.LegendOnCheck,'Value',1)
else
   set(findall(handles.LocationPanel, '-property', 'enable'), 'enable', 'on')
   set(findall(handles.PlotToolsPanel1, '-property', 'enable'), 'enable', 'on')
   set(handles.LegendOnCheck,'Value',0)
end

% --- Executes on button press in DefaultLine.
function DefaultLine_Callback(hObject, eventdata, handles)
% hObject    handle to DefaultLine (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of DefaultLine
DLVal=get(handles.DefaultLine,'Value');
if DLVal==1
   set(handles.text25,'Enable','off')
   set(handles.LineWidthPlot,'Enable','off')
else
   set(handles.text25,'Enable','on')
   set(handles.LineWidthPlot,'Enable','on')
end
% --- Executes on button press in DefaultMarkersPlot.
function DefaultMarkersPlot_Callback(hObject, eventdata, handles)
% hObject    handle to DefaultMarkersPlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of DefaultMarkersPlot
DMVal=get(handles.DefaultMarkersPlot,'Value');
if DMVal==1
   set(handles.text27,'Enable','off')
   set(handles.MarkerSizePlot,'Enable','off')
else
   set(handles.text27,'Enable','on')
   set(handles.MarkerSizePlot,'Enable','on')
end

% --- Executes on button press in DefaultNamePlot.
function DefaultNamePlot_Callback(hObject, eventdata, handles)
% hObject    handle to DefaultNamePlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of DefaultNamePlot
DNVal=get(handles.DefaultNamePlot,'Value');
if DNVal==1
   set(handles.text24,'Enable','off')
   set(handles.LegendName,'Enable','off')
else
   set(handles.text24,'Enable','on')
   set(handles.LegendName,'Enable','on')
end


function LineWidthPlot_Callback(hObject, eventdata, handles)
% hObject    handle to LineWidthPlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of LineWidthPlot as text
%        str2double(get(hObject,'String')) returns contents of LineWidthPlot as a double


% --- Executes during object creation, after setting all properties.
function LineWidthPlot_CreateFcn(hObject, eventdata, handles)
% hObject    handle to LineWidthPlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function MarkerSizePlot_Callback(hObject, eventdata, handles)
% hObject    handle to MarkerSizePlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MarkerSizePlot as text
%        str2double(get(hObject,'String')) returns contents of MarkerSizePlot as a double


% --- Executes during object creation, after setting all properties.
function MarkerSizePlot_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MarkerSizePlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function PlotHeight_Callback(hObject, eventdata, handles)
% hObject    handle to PlotHeight (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of PlotHeight as text
%        str2double(get(hObject,'String')) returns contents of PlotHeight as a double


% --- Executes during object creation, after setting all properties.
function PlotHeight_CreateFcn(hObject, eventdata, handles)
% hObject    handle to PlotHeight (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in LegendLocation.
function LegendLocation_Callback(hObject, eventdata, handles)
% hObject    handle to LegendLocation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns LegendLocation contents as cell array
%        contents{get(hObject,'Value')} returns selected item from LegendLocation


% --- Executes during object creation, after setting all properties.
function LegendLocation_CreateFcn(hObject, eventdata, handles)
% hObject    handle to LegendLocation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on slider movement.
function Position_Callback(hObject, eventdata, handles)
% hObject    handle to Position (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function Position_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Position (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end



function LegendFontSize_Callback(hObject, eventdata, handles)
% hObject    handle to LegendFontSize (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of LegendFontSize as text
%        str2double(get(hObject,'String')) returns contents of LegendFontSize as a double


% --- Executes during object creation, after setting all properties.
function LegendFontSize_CreateFcn(hObject, eventdata, handles)
% hObject    handle to LegendFontSize (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in OffGraphCheck.
function OffGraphCheck_Callback(hObject, eventdata, handles)
% hObject    handle to OffGraphCheck (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of OffGraphCheck


% --- Executes on button press in LegendOnCheck.
function LegendOnCheck_Callback(hObject, eventdata, handles)
% hObject    handle to LegendOnCheck (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of LegendOnCheck


% --- Executes on selection change in SelectDataPlot.
function SelectDataPlot_Callback(hObject, eventdata, handles)
% hObject    handle to SelectDataPlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns SelectDataPlot contents as cell array
%        contents{get(hObject,'Value')} returns selected item from SelectDataPlot
DataSel=get(handles.SelectDataPlot,'Value')-1;
if DataSel ~= 0
    SelCyc=handles.CycleString;
    CycStr=string("Select Cycle:");
    for ee=1:length(SelCyc)
        CycStr(ee+1)=SelCyc(ee);
    end
    CycStr(2:end)=SelCyc(:,DataSel);
    set(handles.SelectCyclePlot,'String',CycStr)
else
   CycStr=string("Select Cycle:");
   set(handles.SelectCyclePlot,'String',CycStr)
end

% --- Executes during object creation, after setting all properties.
function SelectDataPlot_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SelectDataPlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in SelectCyclePlot.
function SelectCyclePlot_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to SelectCyclePlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns SelectCyclePlot contents as cell array
%        contents{get(hObject,'Value')} returns selected item from SelectCyclePlot


% --- Executes during object creation, after setting all properties.
function SelectCyclePlot_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SelectCyclePlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in ColorSelectPlot.
function ColorSelectPlot_Callback(hObject, eventdata, handles)
% hObject    handle to ColorSelectPlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns ColorSelectPlot contents as cell array
%        contents{get(hObject,'Value')} returns selected item from ColorSelectPlot


% --- Executes during object creation, after setting all properties.
function ColorSelectPlot_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ColorSelectPlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in MarkerTypePlot.
function MarkerTypePlot_Callback(hObject, eventdata, handles)
% hObject    handle to MarkerTypePlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns MarkerTypePlot contents as cell array
```

```
%        contents{get(hObject,'Value')} returns selected item from MarkerTypePlot


% --- Executes during object creation, after setting all properties.
function MarkerTypePlot_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MarkerTypePlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in LineTypePlot.
function LineTypePlot_Callback(hObject, eventdata, handles)
% hObject    handle to LineTypePlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns LineTypePlot contents as cell array
%        contents{get(hObject,'Value')} returns selected item from LineTypePlot


% --- Executes during object creation, after setting all properties.
function LineTypePlot_CreateFcn(hObject, eventdata, handles)
% hObject    handle to LineTypePlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



function LegendName_Callback(hObject, eventdata, handles)
% hObject    handle to LegendName (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of LegendName as text
%        str2double(get(hObject,'String')) returns contents of LegendName as a double


% --- Executes during object creation, after setting all properties.
function LegendName_CreateFcn(hObject, eventdata, handles)
% hObject    handle to LegendName (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in SelectFolderButton.
function SelectFolderButton_Callback(hObject, eventdata, handles)
% hObject    handle to SelectFolderButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
source_dir=get(handles.SelectedFolder,'String');

source_files = dir(fullfile(source_dir, '*.xlsx'));
numFiles = length(source_files);
names = cell(numFiles,1);
PMass = nan(numFiles,1);
data = cell(numFiles,1);

% Store variables in handles
handles.source_dir = source_dir;
handles.source_files = source_files;
handles.numFiles = numFiles;
handles.PMass = PMass;
handles.data = data;
guidata(hObject,handles)

% Create cell array of all filenames in the source directory
for ii = 1:numFiles
    names(ii) = {source_files(ii).name(1:end-5)};
end

% Put filenames in DataSet
set(handles.DataSet,'String',names);


function SelectedFolder_Callback(hObject, eventdata, handles)
% hObject    handle to SelectedFolder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of SelectedFolder as text
%        str2double(get(hObject,'String')) returns contents of SelectedFolder as a double


% --- Executes during object creation, after setting all properties.
function SelectedFolder_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SelectedFolder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in BrowseDFButton.
function BrowseDFButton_Callback(hObject, eventdata, handles)
% hObject    handle to BrowseDFButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
DF=get(handles.SelectedFolder,'String');
source_dir = uigetdir(DF);
set(handles.SelectedFolder,'String',(source_dir))

% --- Executes on button press in SQLDataFormat.
function SQLDataFormat_Callback(hObject, eventdata, handles)
% hObject    handle to SQLDataFormat (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of SQLDataFormat
SQLButton = get(handles.SQLDataFormat, 'Value');
if SQLButton == 1
    set(handles.ExcelDataFormat, 'Value', 0);
    set(findall(handles.SQLDataPanel, '-property', 'enable'), 'enable', 'on')
    set(findall(handles.ExcelDataPanel, '-property', 'enable'), 'enable', 'off')
else
    set(handles.ExcelDataFormat, 'Value', 1);
    set(findall(handles.SQLDataPanel, '-property', 'enable'), 'enable', 'off')
    set(findall(handles.ExcelDataPanel, '-property', 'enable'), 'enable', 'on')
end

% --- Executes on button press in ExcelDataFormat.
function ExcelDataFormat_Callback(hObject, eventdata, handles)
% hObject    handle to ExcelDataFormat (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of ExcelDataFormat
SQLButton = get(handles.SQLDataFormat, 'Value');
if SQLButton == 1
    set(handles.ExcelDataFormat, 'Value', 0);
    set(findall(handles.SQLDataPanel, '-property', 'enable'), 'enable', 'on')
    set(findall(handles.ExcelDataPanel, '-property', 'enable'), 'enable', 'off')
else
    set(handles.ExcelDataFormat, 'Value', 1);
    set(findall(handles.SQLDataPanel, '-property', 'enable'), 'enable', 'off')
    set(findall(handles.ExcelDataPanel, '-property', 'enable'), 'enable', 'on')
end


% --- Executes on button press in AutoP2.
function AutoP2_Callback(hObject, eventdata, handles)
% hObject    handle to AutoP2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of AutoP2
Auto=get(handles.AutoP2,'Value');
if Auto==1
    set(handles.MinRangeP,'Enable','off')
else
    set(handles.MinRangeP,'Enable','on')
end


% --- Executes on button press in plotButton.
function plotButton_Callback(hObject, eventdata, handles)
% hObject    handle to plotButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
selectedValue = get(handles.DataSet, 'Value');
XAxis1=handles.XAxis1;
XAxis2=handles.XAxis2;
XAxisA=handles.XAxis;
YAxis1=handles.YAxis1;
YAxis2=handles.YAxis2;
YSAxis1=handles.YSAxis1;
YSAxis2=handles.YSAxis2;
cla reset
for gg=1:length(selectedValue)
    XAxisRaw=[];
    XAxisRawA=[];
    XAxisRawB=[];
    YAxisRawA=[];
    YAxisRawB=[];
    YAxisRawAS=[];
    YAxisRawBS=[];
```

```
Cycles=handles.cycles(gg);
XAxisB=XAxisA{gg};
XAxisC=XAxis1{gg};
XAxisD=XAxis2{gg};
if isempty(XAxisB)==1
else
    XAxisRaw=XAxisB;
    YAxisA=YAxis1{gg};
    YAxisB=YAxis2{gg};
    YAxisRawA=YAxisA;
    YAxisRawB=YAxisB;
    figure(1)
    yyaxis left
    plot(XAxisB,YAxisA,'b-')
    hold on
    if isempty(YAxisB)
    else
    yyaxis left
    plot(XAxisB,YAxisB,'b-')
    hold on
    end
    if isempty(YSAxis1)
    else
    YSAxisA=YSAxis1{gg};
    YAxisRawAS=YAxisA;
    figure(1)
    yyaxis right
    plot(XAxisB,YSAxisA,'r-')
    hold on
    end
    if isempty(YSAxis2)
    else
    YSAxisB=YSAxis2{gg};
    YAXisRawBS=YAxisB;
    figure(1)
    yyaxis right
    plot(XAxisB,YSAxisB,'r-')
    hold on
    end
end
if isempty(XAxisC)==1
else
    XAxis=XAxis1{gg};
    YAxis=YAxis1{gg};
    [R C]= size(XAxis);
    bb=1;
    cc=1;
    for aa=1:Cycles
        if isnan(XAxis(:,aa))
        else
            XAxisRawA(:,bb)=XAxis(:,aa);
```

```
            YAxisRawA(:,bb)=YAxis(:,aa);
            figure(1)
            yyaxis left
            plot(XAxis(:,aa),YAxis(:,aa),'b-')
            hold on
            bb=bb+1
        end
    end
    XAxis=XAxis1{gg};
    if isempty(YSAxis1)
    else
    YSAxis=YSAxis1{gg};
    [R C]= size(XAxis);
    for aa=1:Cycles
        if isnan(XAxis(:,aa))
        else
            YAxisRawAS(:,cc)=YSAxis(:,aa);
            figure(1)
            yyaxis right
            plot(XAxis(:,aa),YSAxis(:,aa),'r-')
            hold on
            cc=cc+1;
        end
    end
    end
end
if isempty(XAxisD)==1
else
    XAxis=XAxis2{gg};
    if isempty(YAxis2)
    else
    YAxis=YAxis2{gg};
    [R C]= size(XAxis);
    bb=1;
    cc=1;
    for aa=1:Cycles
        if isnan(XAxis(:,aa))
        else
            XAxisRawB(:,bb)=XAxis(:,aa);
            YAxisRawB(:,bb)=YAxis(:,aa);
            figure(1)
            yyaxis left
            plot(XAxis(:,aa),YAxis(:,aa),'b-')
            hold on
            bb=bb+1;
        end
    end
    end
    if isempty(YSAxis2)
    else
    XAxis=XAxis2{gg};
```

```
        YSAxis=YSAxis2{gg};
        [R C]= size(XAxis);
        for aa=1:Cycles
            if isnan(XAxis(:,aa))
            else
                YAxisRawBS(:,cc)=YSAxis(:,aa);
                figure(1)
                yyaxis right
                plot(XAxis(:,aa),YSAxis(:,aa),'r-')
                hold on
                cc=cc+1;
            end
        end
        end
    end
    handles.XAxisRaw{gg}=XAxisRaw
    handles.XAxisRawA{gg}=XAxisRawA
    handles.XAxisRawB{gg}=XAxisRawB
    handles.YAxisRawA{gg}=YAxisRawA
    handles.YAxisRawB{gg}=YAxisRawB
    handles.YAxisRawAS{gg}=YAxisRawAS
    handles.YAxisRawBS{gg}=YAxisRawBS
end
guidata(hObject,handles);
XAVal=handles.XAValue;
YAVal=handles.YAValue;
if XAVal == 2
    xlabel('Time (s)')
elseif XAVal ==3
    xlabel('Voltage (V)')
elseif XAVal == 4
    xlabel('Cycles')
elseif XAVal == 5
    xlabel('Capacity (mAh)')
elseif XAVal == 6
    xlabel('Specific Capacity (mAh/g)')
end
if YAVal == 2
    yyaxis left
    ylabel('Voltage (V)')
elseif YAVal == 3
    yyaxis left
    ylabel('Current (A)')
elseif YAVal == 4
    yyaxis left
    ylabel('Capacity (mAh)')
elseif YAVal == 5
    yyaxis left
    ylabel('Specific Capacity (mAh/g)')
elseif YAVal == 6
    ylabel('dQ/dV (mAh/V)')
```

```
elseif YAVal == 7
    yyaxis left
    ylabel('Columbic Efficency (%)')
end
if isempty(handles.YSAValue)
else
YSAVal=handles.YSAValue;
if YSAVal == 2
    yyaxis right
    ylabel('Voltage (V)')
elseif YSAVal == 3
    yyaxis right
    ylabel('Current (A)')
elseif YSAVal == 4
    yyaxis right
    ylabel('Capacity (mAh)')
elseif YSAVal == 5
    yyaxis right
    ylabel('Specific Capacity (mAh/g)')
elseif YSAVal == 6
    ylabel('dQ/dV (mAh/V)')
elseif YSAVal == 7
    yyaxis right
    ylabel('Columbic Efficency (%)')
end
end

% --- Executes on button press in DefaultSizePlot.
function DefaultSizePlot_Callback(hObject, eventdata, handles)
% hObject    handle to DefaultSizePlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of DefaultSizePlot
DSVal=get(handles.DefaultSizePlot,'Value');
if DSVal==1
    set(handles.text28,'Enable','off')
    set(handles.text30,'Enable','off')
    set(handles.PlotWidth,'Enable','off')
    set(handles.PlotHeight,'Enable','off')
else
    set(handles.text28,'Enable','on')
    set(handles.text30,'Enable','on')
    set(handles.PlotHeight,'Enable','on')
    set(handles.PlotWidth,'Enable','on')
end

% --- Executes on button press in ChargingData.
function ChargingData_Callback(hObject, eventdata, handles)
% hObject    handle to ChargingData (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ChargingData


% --- Executes on button press in DischargingData.
function DischargingData_Callback(hObject, eventdata, handles)
% hObject    handle to DischargingData (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of DischargingData


% --- Executes on button press in SelectXAxis.
function SelectXAxis_Callback(hObject, eventdata, handles)
% hObject    handle to SelectXAxis (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in SelectYAxis.
XAVal=get(handles.XAxisDataType,'Value');
ChVal=get(handles.ChargingData,'Value');
DisVal=get(handles.DischargingData,'Value');
ContVal=get(handles.ContiniousButton,'Value');
CycVal=get(handles.CyclicButton,'Value');
IndVal=get(handles.IndividualButton,'Value');
AllVal=get(handles.AllButton,'Value');
selectedValue = get(handles.DataSet, 'Value');
selectedData=get(handles.DataSet, 'String');
DataStr(1)=string("Select Data Set:");
ee=2;
for dd=selectedValue
DataStr(ee)=selectedData{dd};
ee=ee+1;
end
set(handles.SelectDataPlot,'String',DataStr);
set(handles.RawDataSet,'String',DataStr);
XAxisA=[];
XAxis1=[];
XAxis2=[];
str=string("");
for gg=1:length(selectedValue)
   Cycles=handles.cycles(gg);
if AllVal==1
   for cc=1:Cycles
      str(cc,gg)=string(['Cycles ' num2str(cc)]);
   end
   if ChVal==0 && DisVal==0
      if XAVal == 1
         errordlg('Please Select a Valid X Axis Data Type','Plot Error');
      elseif XAVal == 2
```

```
        errordlg('Please Select a Valid X Axis Data Type','Plot Error');
    elseif XAVal == 3
        errordlg('Please Select a Valid X Axis Data Type','Plot Error');
    elseif XAVal == 4
        XAxisA=1:Cycles;
    elseif XAVal == 5
        errordlg('Please Select a Valid X Axis Data Type','Plot Error');
    elseif XAVal == 6
        errordlg('Please Select a Valid X Axis Data Type','Plot Error');
    end
elseif ChVal==1 && DisVal==1
    if XAVal == 1
        errordlg('Please Select a Valid Plot Type','Plot Error');
    elseif XAVal == 2
        if ContVal == 1
            XAxis1=cell2mat(handles.ChTime(gg));
            XAxis2=cell2mat(handles.DisTime(gg));
        elseif CycVal == 1
            XAxis1=cell2mat(handles.ChCycTime(gg));
            XAxis2=cell2mat(handles.DisCycTime(gg));
        elseif IndVal == 1
            XAxis1=cell2mat(handles.ChCycTime(gg));
            XAxis2=cell2mat(handles.DisCycTime(gg));
        end
    elseif XAVal == 3
        XAxis1=cell2mat(handles.ChVolt(gg));
        XAxis2=cell2mat(handles.DisVolt(gg));
    elseif XAVal == 4
        errordlg('Please Select a Valid Plot Type','Plot Error');
    elseif XAVal == 5
        XAxis1=cell2mat(handles.ChCap(gg));
        XAXis2=cell2mat(handles.DisCap(gg));
    elseif XAVal == 6
        XAxis1=cell2mat(handles.chSpecCap(gg));
        XAXis2=cell2mat(handles.disSpecCap(gg));
    end
elseif ChVal==1 && DisVal==0
    if XAVal == 1
        errordlg('Please Select a Valid Plot Type','Plot Error');
    elseif XAVal == 2
        if ContVal == 1
            XAxis1=cell2mat(handles.ChTime(gg));
        elseif CycVal == 1
            XAxis1=cell2mat(handles.ChCycTime(gg));
        elseif IndVal == 1
            XAxis1=cell2mat(handles.ChCycTime(gg));
        end
    elseif XAVal == 3
        XAxis1=cell2mat(handles.ChVolt(gg));
    elseif XAVal == 4
        errordlg('Please Select a Valid Plot Type','Plot Error');
```

```
        elseif XAVal == 5
            XAxis1=cell2mat(handles.ChCap(gg));
        elseif XAVal == 6
            XAxis1=cell2mat(handles.chSpecCap(gg));
        end
    elseif ChVal==0 && DisVal==1
        if XAVal == 1
            errordlg('Please Select a Valid Plot Type','Plot Error');
        elseif XAVal == 2
            if ContVal == 1
                XAxis2=cell2mat(handles.DisTime(gg));
            elseif CycVal == 1
                XAxis2=cell2mat(handles.DisCycTime(gg));
            elseif IndVal == 1
                XAxis2=cell2mat(handles.DisCycTime(gg));
            end
        elseif XAVal == 3
            XAxis2=handles.DisVolt;
        elseif XAVal == 4
            errordlg('Please Select a Valid Plot Type','Plot Error');
        elseif XAVal == 5
            XAxis1=cell2mat(handles.ChCap(gg));
            XAXis2=cell2mat(handles.DisCap(gg));
        elseif XAVal == 6
            XAxis1=cell2mat(handles.chSpecCap(gg));
            XAXis2=cell2mat(handles.disSpecCap(gg));
        end
    end
else
    str2=get(handles.CyclesToPlot,'String');
    [indNums groupNums]=CycleNums(str2);
    if isempty(get(handles.CyclesToPlot, 'String')) ...
            || isempty(indNums) && isempty(groupNums) ...
            || ~isempty(groupNums) && max(groupNums) > Cycles ...
            || ~isempty(indNums) && max(indNums) > Cycles ...
            || ~isempty(groupNums) && mod(length(groupNums), 2) ~= 0 ...
            || ~isempty(groupNums) && groupNums(1) > groupNums(2)
        errordlg('Please enter a valid cycle range','Cycle Number Error');
        return
    else
        ee=1;
        for aa=indNums
            str(ee,gg)=string(['Cycle ' num2str(aa)]);
            ee=ee+1;
        end
        for bb=1:2:length(groupNums)
            groupNum=groupNums(bb):groupNums(bb+1);
            for aa=groupNum
                str(aa,gg)=string(['Cycle ' num2str(aa)]);
                ee=ee+1;
            end
```

```
    end
    if ChVal==0 && DisVal==0
    if XAVal == 1
        errordlg('Please Select a Valid X Axis Data Type','Plot Error');
    elseif XAVal == 2
        errordlg('Please Select a Valid X Axis Data Type','Plot Error');
    elseif XAVal == 3
        errordlg('Please Select a Valid X Axis Data Type','Plot Error');
    elseif XAVal == 4
        errordlg('Please Select a Valid X Axis Data Type','Plot Error');
    elseif XAVal == 5
        errordlg('Please Select a Valid X Axis Data Type','Plot Error');
    elseif XAVal == 6
        errordlg('Please Select a Valid X Axis Data Type','Plot Error');
    end
elseif ChVal==1 && DisVal==1
    if XAVal == 1
        errordlg('Please Select a Valid X Axis Data Type','Plot Error');
    elseif XAVal == 2
        if ContVal == 1
            XAxis=cell2mat(handles.ChTime(gg));
            XAxis1=nan(size(XAxis));
            for aa=indNums
                XAxis1(:,aa)=XAxis(:,aa);
            end
            for bb=1:2:length(groupNums)
                groupNum=groupNums(bb):groupNums(bb+1);
            for aa=groupNum
                XAxis1(:,aa)=XAxis(:,aa);
            end
            end
            XAxis=cell2mat(handles.DisTime(gg));
            XAxis2=nan(size(XAxis));
            for aa=indNums
                XAxis2(:,aa)=XAxis(:,aa);
            end
            for bb=1:2:length(groupNums)
                groupNum=groupNums(bb):groupNums(bb+1);
            for aa=groupNum
                XAxis2(:,aa)=XAxis(:,aa);
            end
            end
        elseif CycVal == 1
            XAxis=cell2mat(handles.ChCycTime(gg));
            XAxis1=nan(size(XAxis));
            for aa=indNums
                XAxis1(:,aa)=XAxis(:,aa);
            end
            for bb=1:2:length(groupNums)
                groupNum=groupNums(bb):groupNums(bb+1);
            for aa=groupNum
```

```
            XAxis1(:,aa)=XAxis(:,aa);
        end
        end
        XAxis=cell2mat(handles.DisCycTime(gg));
        XAxis2=nan(size(XAxis));
        for aa=indNums
            XAxis2(:,aa)=XAxis(:,aa);
        end
        for bb=1:2:length(groupNums)
            groupNum=groupNums(bb):groupNums(bb+1);
        for aa=groupNum
            XAxis2(:,aa)=XAxis(:,aa);
        end
        end
    elseif IndVal == 1
        XAxis=cell2mat(handles.ChCycTime(gg));
        XAxis1=nan(size(XAxis));
        for aa=indNums
            XAxis1(:,aa)=XAxis(:,aa);
        end
        for bb=1:2:length(groupNums)
            groupNum=groupNums(bb):groupNums(bb+1);
        for aa=groupNum
            XAxis1(:,aa)=XAxis(:,aa);
        end
        end
        XAxis=cell2mat(handles.DisCycTime(gg));
        XAxis2=nan(size(XAxis));
        for aa=indNums
            XAxis2(:,aa)=XAxis(:,aa);
        end
        for bb=1:2:length(groupNums)
            groupNum=groupNums(bb):groupNums(bb+1);
        for aa=groupNum
            XAxis2(:,aa)=XAxis(:,aa);
        end
        end
    end
elseif XAVal == 3
    XAxis=cell2mat(handles.ChVolt(gg));
    XAxis1=nan(size(XAxis));
        for aa=indNums
            XAxis1(:,aa)=XAxis(:,aa);
        end
        for bb=1:2:length(groupNums)
            groupNum=groupNums(bb):groupNums(bb+1);
        for aa=groupNum
            XAxis1(:,aa)=XAxis(:,aa);
        end
        end
    XAxis=cell2mat(handles.DisVolt(gg));
```

```
    XAxis2=nan(size(XAxis));
        for aa=indNums
            XAxis2(:,aa)=XAxis(:,aa);
        end
        for bb=1:2:length(groupNums)
            groupNum=groupNums(bb):groupNums(bb+1);
        for aa=groupNum
            XAxis2(:,aa)=XAxis(:,aa);
        end
        end
elseif XAVal == 4
    errordlg('Please Select a Valid X Axis Data Type','Plot Error');
elseif XAVal == 5
    XAxis=cell2mat(handles.ChCap(gg));
    XAxis1=nan(size(XAxis));
        for aa=indNums
            XAxis1(:,aa)=XAxis(:,aa);
        end
        for bb=1:2:length(groupNums)
            groupNum=groupNums(bb):groupNums(bb+1);
        for aa=groupNum
            XAxis1(:,aa)=XAxis(:,aa);
        end
        end
    XAxis=cell2mat(handles.DisCap(gg));
    XAxis2=nan(size(XAxis));
        for aa=indNums
            XAxis2(:,aa)=XAxis(:,aa);
        end
        for bb=1:2:length(groupNums)
            groupNum=groupNums(bb):groupNums(bb+1);
        for aa=groupNum
            XAxis2(:,aa)=XAxis(:,aa);
        end
        end
elseif XAVal == 6
    XAxis=cell2mat(handles.chSpecCap(gg));
    XAxis1=nan(size(XAxis));
        for aa=indNums
            XAxis1(:,aa)=XAxis(:,aa);
        end
        for bb=1:2:length(groupNums)
            groupNum=groupNums(bb):groupNums(bb+1);
        for aa=groupNum
            XAxis1(:,aa)=XAxis(:,aa);
        end
        end
    XAxis=cell2mat(handles.disSpecCap(gg));
    XAxis2=nan(size(XAxis));
        for aa=indNums
            XAxis2(:,aa)=XAxis(:,aa);
```

```
            end
            for bb=1:2:length(groupNums)
               groupNum=groupNums(bb):groupNums(bb+1);
            for aa=groupNum
               XAxis2(:,aa)=XAxis(:,aa);
            end
            end
      end
   elseif ChVal==1 && DisVal==0
      if XAVal == 1
         errordlg('Please Select a Valid X Axis Data Type','Plot Error');
      elseif XAVal == 2
         if ContVal == 1
            XAxis=cell2mat(handles.ChTime(gg));
            XAxis1=nan(size(XAxis));
            for aa=indNums
               XAxis1(:,aa)=XAxis(:,aa);
            end
            for bb=1:2:length(groupNums)
               groupNum=groupNums(bb):groupNums(bb+1);
            for aa=groupNum
               XAxis1(:,aa)=XAxis(:,aa);
            end
            end
         elseif CycVal == 1
            XAxis=cell2mat(handles.ChCycTime(gg));
            XAxis1=nan(size(XAxis));
            for aa=indNums
               XAxis1(:,aa)=XAxis(:,aa);
            end
            for bb=1:2:length(groupNums)
               groupNum=groupNums(bb):groupNums(bb+1);
            for aa=groupNum
               XAxis1(:,aa)=XAxis(:,aa);
            end
            end
         elseif IndVal == 1
            XAxis=cell2mat(handles.ChCycTime(gg));
            XAxis1=nan(size(XAxis));
            for aa=indNums
               XAxis1(:,aa)=XAxis(:,aa);
            end
            for bb=1:2:length(groupNums)
               groupNum=groupNums(bb):groupNums(bb+1);
            for aa=groupNum
               XAxis1(:,aa)=XAxis(:,aa);
            end
            end
         end
      elseif XAVal == 3
         XAxis=cell2mat(handles.ChVolt(gg));
```

```
        XAxis1=nan(size(XAxis));
            for aa=indNums
                XAxis1(:,aa)=XAxis(:,aa);
            end
            for bb=1:2:length(groupNums)
                groupNum=groupNums(bb):groupNums(bb+1);
            for aa=groupNum
                XAxis1(:,aa)=XAxis(:,aa);
            end
            end
    elseif XAVal == 4
        errordlg('Please Select a Valid X Axis Data Type','Plot Error');
    elseif XAVal == 5
        XAxis=cell2mat(handles.ChCap(gg));
        XAxis1=nan(size(XAxis));
            for aa=indNums
                XAxis1(:,aa)=XAxis(:,aa);
            end
            for bb=1:2:length(groupNums)
                groupNum=groupNums(bb):groupNums(bb+1);
            for aa=groupNum
                XAxis1(:,aa)=XAxis(:,aa);
            end
            end
    elseif XAVal == 6
        XAxis=cell2mat(handles.chSpecCap(gg));
        XAxis1=nan(size(XAxis));
            for aa=indNums
                XAxis1(:,aa)=XAxis(:,aa);
            end
            for bb=1:2:length(groupNums)
                groupNum=groupNums(bb):groupNums(bb+1);
            for aa=groupNum
                XAxis1(:,aa)=XAxis(:,aa);
            end
            end
    end
elseif ChVal==0 && DisVal==1
    if XAVal == 1
        errordlg('Please Select a Valid X Axis Data Type','Plot Error');
    elseif XAVal == 2
        if ContVal == 1
            XAxis=cell2mat(handles.DisTime(gg));
            XAxis2=nan(size(XAxis));
            for aa=indNums
                XAxis2(:,aa)=XAxis(:,aa);
            end
            for bb=1:2:length(groupNums)
                groupNum=groupNums(bb):groupNums(bb+1);
            for aa=groupNum
                XAxis2(:,aa)=XAxis(:,aa);
```

```
        end
        end
    elseif CycVal == 1
        XAxis=cell2mat(handles.DisCycTime(gg));
        XAxis2=nan(size(XAxis));
        for aa=indNums
            XAxis2(:,aa)=XAxis(:,aa);
        end
        for bb=1:2:length(groupNums)
            groupNum=groupNums(bb):groupNums(bb+1);
        for aa=groupNum
            XAxis2(:,aa)=XAxis(:,aa);
        end
        end
    elseif IndVal == 1
        XAxis=cell2mat(handles.DisCycTime(gg));
        XAxis2=nan(size(XAxis));
        for aa=indNums
            XAxis2(:,aa)=XAxis(:,aa);
        end
        for bb=1:2:length(groupNums)
            groupNum=groupNums(bb):groupNums(bb+1);
        for aa=groupNum
            XAxis2(:,aa)=XAxis(:,aa);
        end
        end
    end
elseif XAVal == 3
    XAxis=cell2mat(handles.DisVolt(gg));
    XAxis2=nan(size(XAxis));
        for aa=indNums
            XAxis2(:,aa)=XAxis(:,aa);
        end
        for bb=1:2:length(groupNums)
            groupNum=groupNums(bb):groupNums(bb+1);
        for aa=groupNum
            XAxis2(:,aa)=XAxis(:,aa);
        end
        end
elseif XAVal == 4
    errordlg('Please Select a Valid X Axis Data Type','Plot Error');
elseif XAVal ==5
    XAxis=cell2mat(handles.DisCap(gg));
    XAxis2=nan(size(XAxis));
        for aa=indNums
            XAxis2(:,aa)=XAxis(:,aa);
        end
        for bb=1:2:length(groupNums)
            groupNum=groupNums(bb):groupNums(bb+1);
        for aa=groupNum
            XAxis2(:,aa)=XAxis(:,aa);
```

```
            end
            end
        elseif XAVal == 6
            XAxis=cell2mat(handles.disSpecCap(gg));
            XAxis2=nan(size(XAxis));
            for aa=indNums
                XAxis2(:,aa)=XAxis(:,aa);
            end
            for bb=1:2:length(groupNums)
                groupNum=groupNums(bb):groupNums(bb+1);
            for aa=groupNum
                XAxis2(:,aa)=XAxis(:,aa);
            end
            end
        end
        end
    end
end
    handles.XAxis{gg}=XAxisA;
    handles.XAxis1{gg}=XAxis1;
    handles.XAxis2{gg}=XAxis2;
end
handles.XAValue=XAVal;
handles.CycleString=str;
guidata(hObject,handles)

function SelectYAxis_Callback(hObject, eventdata, handles)
% hObject    handle to SelectYAxis (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
YAVal=get(handles.YAxisDataType1,'Value');
XAVal=get(handles.XAxisDataType,'Value');
ChVal=get(handles.ChargingData,'Value');
DisVal=get(handles.DischargingData,'Value');
ContVal=get(handles.ContiniousButton,'Value');
CycVal=get(handles.CyclicButton,'Value');
IndVal=get(handles.IndividualButton,'Value');
selectedValue = get(handles.DataSet, 'Value');
YAxis1=[];
YAxis2=[];
for gg=1:length(selectedValue)
    if ChVal==0 && DisVal==0
        if YAVal == 1
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        elseif YAVal == 2
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        elseif YAVal == 3
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        elseif YAVal == 4
            YAxis1=cell2mat(handles.chCapData(gg));
            YAxis2=cell2mat(handles.disCapData(gg));
```

```
        elseif YAVal == 5
            YAxis1=cell2mat(handles.chSpecCapData(gg));
            YAxis2=cell2mat(handles.disSpecCapData(gg));
        elseif YAVal == 6
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        elseif YAVal == 7
            YAxis1=cell2mat(handles.coulEff);
        end

    elseif ChVal==1 && DisVal==1
        if YAVal == 1
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        elseif YAVal == 2
            YAxis1=cell2mat(handles.ChVolt(gg));
            YAxis2=cell2mat(handles.DisVolt(gg));
        elseif YAVal == 3
            YAxis1=cell2mat(handles.ChCurrent(gg));
            YAxis2=cell2mat(handles.DisCurrent(gg));
        elseif YAVal == 4
            YAxis1=cell2mat(handles.ChCap(gg));
            YAxis2=cell2mat(handles.DisCap(gg));
        elseif YAVal == 5
            YAxis1=cell2mat(handles.chSpecCap(gg));
            YAxis2=cell2mat(handles.disSpecCap(gg));
        elseif YAVal == 6
            ChVolt=cell2mat(handles.ChVolt(gg));
            DisVolt=cell2mat(handles.DisVolt(gg));
            ChCap=cell2mat(handles.ChCap(gg));
            DisCap=cell2mat(handles.DisCap(gg));
            chdQdV=cell2mat(handles.chdQdV(gg));
            disdQdV=cell2mat(handles.disdQdV(gg));
            [rows cycles]=size(chdQdV);
            % Get dQ/dV Data
            for zzz = 1:cycles
                for jj = 2:rows
                    chdQdV(jj-1,zzz) = (ChCap(jj,zzz)-ChCap(jj-1,zzz))/(ChVolt(jj,zzz)-ChVolt(jj-1,zzz));
                end
            end
            [rows cycles]=size(disdQdV);
            for zzz = 1:cycles
                for jj = 2:rows
                    disdQdV(jj-1,zzz) = (DisCap(jj,zzz)-DisCap(jj-1,zzz))/(DisVolt(jj,zzz)-DisVolt(jj-
1,zzz));
                end
            end
            YAxis1=chdQdV;
            YAxis2=disdQdV;
        elseif YAVal == 7
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        end
    elseif ChVal==1 && DisVal==0
```

```
        if YAVal == 1
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        elseif YAVal == 2
            YAxis1=cell2mat(handles.ChVolt(gg));
        elseif YAVal == 3
            YAxis1=cell2mat(handles.ChCurrent(gg));
        elseif YAVal == 4
            YAxis1=cell2mat(handles.ChCap(gg));
        elseif YAVal == 5
            YAxis1=cell2mat(handles.chSpecCap(gg));
        elseif YAVal == 6
            ChVolt=cell2mat(handles.ChVolt(gg));
            ChCap=cell2mat(handles.ChCap(gg));
            chdQdV=cell2mat(handles.chdQdV(gg));
            [rows cycles]=size(chdQdV);
            % Get dQ/dV Data
            for zzz = 1:cycles
                for jj = 2:rows
                    chdQdV(jj-1,zzz) = (ChCap(jj,zzz)-ChCap(jj-1,zzz))/(ChVolt(jj,zzz)-ChVolt(jj-1,zzz));
                end
            end
            YAxis1=chdQdV;
        elseif YAVal == 7
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        end
    elseif ChVal==0 && DisVal==1
        if YAVal == 1
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        elseif YAVal == 2
            YAxis2=cell2mat(handles.DisVolt(gg));
        elseif YAVal == 3
            YAxis2=cell2mat(handles.DisCurrent(gg));
        elseif YAVal == 4
            YAxis2=cell2mat(handles.DisCap(gg));
        elseif YAVal == 5
            YAxis2=cell2mat(handles.disSpecCap(gg));
        elseif YAVal == 6
            DisVolt=cell2mat(handles.DisVolt(gg));
            DisCap=cell2mat(handles.DisCap(gg));
            disdQdV=cell2mat(handles.disdQdV(gg));
            [rows cycles]=size(disdQdV);
            % Get dQ/dV Data
            for zzz = 1:cycles
                for jj = 2:rows
                    disdQdV(jj-1,zzz) = (DisCap(jj,zzz)-DisCap(jj-1,zzz))/(DisVolt(jj,zzz)-DisVolt(jj-
1,zzz));
                end
            end
            YAxis2=disdQdV;
        elseif YAVal == 7
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
```

```
        end
    end
    handles.YAxis1{gg}=YAxis1;
    handles.YAxis2{gg}=YAxis2;
end
handles.YSAxis1={};
handles.YSAxis2={};
handles.YSAValue=[];
handles.YAValue=YAVal;
guidata(hObject,handles);


% --- Executes on button press in SelectSYAxis.
function SelectSYAxis_Callback(hObject, eventdata, handles)
% hObject    handle to SelectSYAxis (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
YAVal=get(handles.YAxisDataType2,'Value');
XAVal=get(handles.XAxisDataType,'Value');
ChVal=get(handles.ChargingData,'Value');
DisVal=get(handles.DischargingData,'Value');
ContVal=get(handles.ContiniousButton,'Value');
CycVal=get(handles.CyclicButton,'Value');
IndVal=get(handles.IndividualButton,'Value');
selectedValue = get(handles.DataSet, 'Value');
YAxis1=[];
YAxis2=[];
for gg=1:length(selectedValue)
    if ChVal==0 && DisVal==0
        if YAVal == 1
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        elseif YAVal == 2
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        elseif YAVal == 3
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        elseif YAVal == 4
            YAxis1=cell2mat(handles.chCapData(gg));
            YAxis2=cell2mat(handles.disCapData(gg));
        elseif YAVal == 5
            YAxis1=cell2mat(handles.chSpecCapData(gg));
            YAxis2=cell2mat(handles.disSpecCapData(gg));
        elseif YAVal == 6
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        elseif YAVal == 7
            YAxis1=cell2mat(handles.coulEff);
        end

    elseif ChVal==1 && DisVal==1
        if YAVal == 1
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        elseif YAVal == 2
```

```
            YAxis1=cell2mat(handles.ChVolt(gg));
            YAxis2=cell2mat(handles.DisVolt(gg));
        elseif YAVal == 3
            YAxis1=cell2mat(handles.ChCurrent(gg));
            YAxis2=cell2mat(handles.DisCurrent(gg));
        elseif YAVal == 4
            YAxis1=cell2mat(handles.ChCap(gg));
            YAxis2=cell2mat(handles.DisCap(gg));
        elseif YAVal == 5
            YAxis1=cell2mat(handles.chSpecCap(gg));
            YAxis2=cell2mat(handles.disSpecCap(gg));
        elseif YAVal == 6
            ChVolt=cell2mat(handles.ChVolt(gg));
            DisVolt=cell2mat(handles.DisVolt(gg));
            ChCap=cell2mat(handles.ChCap(gg));
            DisCap=cell2mat(handles.DisCap(gg));
            chdQdV=cell2mat(handles.chdQdV(gg));
            disdQdV=cell2mat(handles.disdQdV(gg));
            [rows cycles]=size(chdQdV);
            % Get dQ/dV Data
            for zzz = 1:cycles
                for jj = 2:rows
                    chdQdV(jj-1,zzz) = (ChCap(jj,zzz)-ChCap(jj-1,zzz))/(ChVolt(jj,zzz)-ChVolt(jj-1,zzz));
                end
            end
            [rows cycles]=size(disdQdV);
            for zzz = 1:cycles
                for jj = 2:rows
                    disdQdV(jj-1,zzz) = (DisCap(jj,zzz)-DisCap(jj-1,zzz))/(DisVolt(jj,zzz)-DisVolt(jj-
1,zzz));
                end
            end
            YAxis1=chdQdV;
            YAxis2=disdQdV;
        elseif YAVal == 7
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        end
    elseif ChVal==1 && DisVal==0
        if YAVal == 1
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        elseif YAVal == 2
            YAxis1=cell2mat(handles.ChVolt(gg));
        elseif YAVal == 3
            YAxis1=cell2mat(handles.ChCurrent(gg));
        elseif YAVal == 4
            YAxis1=cell2mat(handles.ChCap(gg));
        elseif YAVal == 5
            YAxis1=cell2mat(handles.chSpecCap(gg));
        elseif YAVal == 6
            ChVolt=cell2mat(handles.ChVolt(gg));
            ChCap=cell2mat(handles.ChCap(gg));
```

```
            chdQdV=cell2mat(handles.chdQdV(gg));
            [rows cycles]=size(chdQdV);
            % Get dQ/dV Data
            for zzz = 1:cycles
                for jj = 2:rows
                    chdQdV(jj-1,zzz) = (ChCap(jj,zzz)-ChCap(jj-1,zzz))/(ChVolt(jj,zzz)-ChVolt(jj-1,zzz));
                end
            end
            YAxis1=chdQdV;
        elseif YAVal == 7
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        end
    elseif ChVal==0 && DisVal==1
        if YAVal == 1
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        elseif YAVal == 2
            YAxis2=cell2mat(handles.DisVolt(gg));
        elseif YAVal == 3
            YAxis2=cell2mat(handles.DisCurrent(gg));
        elseif YAVal == 4
            YAxis2=cell2mat(handles.DisCap(gg));
        elseif YAVal == 5
            YAxis2=cell2mat(handles.disSpecCap(gg));
        elseif YAVal == 6
            DisVolt=cell2mat(handles.DisVolt(gg));
            DisCap=cell2mat(handles.DisCap(gg));
            disdQdV=cell2mat(handles.disdQdV(gg));
            [rows cycles]=size(disdQdV);
            % Get dQ/dV Data
            for zzz = 1:cycles
                for jj = 2:rows
                    disdQdV(jj-1,zzz) = (DisCap(jj,zzz)-DisCap(jj-1,zzz))/(DisVolt(jj,zzz)-DisVolt(jj-
1,zzz));
                end
            end
            YAxis2=disdQdV;
        elseif YAVal == 7
            errordlg('Please Select a Valid Y Axis Data Type','Plot Error');
        end
    end
    handles.YSAxis1{gg}=YAxis1;
    handles.YSAxis2{gg}=YAxis2;
end
handles.YSAValue=YAVal;
guidata(hObject,handles);


% --- Executes on button press in RawData.
function RawData_Callback(hObject, eventdata, handles)
% hObject    handle to RawData (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
RawDataVal=get(handles.RawDataSet,'Value')-1;
if RawDataVal==0
else
        XAxisRaw=handles.XAxisRaw{RawDataVal};
        XAxisRawA=handles.XAxisRawA{RawDataVal};
        XAxisRawB=handles.XAxisRawB{RawDataVal};
        YAxisRawA=handles.YAxisRawA{RawDataVal};
        YAxisRawB=handles.YAxisRawB{RawDataVal};
        YAxisRawAS=handles.YAxisRawAS{RawDataVal};
        YAxisRawBS=handles.YAxisRawBS{RawDataVal};
        SelCyc=handles.CycleString(:,RawDataVal);
        XAVal=handles.XAValue;
        YAVal=handles.YAValue;
        YASVal=handles.YSAValue;
        XAStr=get(handles.XAxisDataType,'String');
        YAStr=get(handles.YAxisDataType1,'String');
        YASStr=get(handles.YAxisDataType2,'String');
        XASel=XAStr(XAVal);
        YASel=YAStr(YAVal);
        YASSel=YASStr(YASVal);

        if isempty(XAxisRaw)
            [R C]=size(XAxisRawA);
            RawData=NaN(R,6*C);
            ColLabel=strings(6*C,1);
            for aa=1:C
                ColLabel(6*aa-5)=strcat("Charging ", SelCyc(aa), " ", XASel, ' Data');
                ColLabel(6*aa-4)=strcat("Charging ", SelCyc(aa), " ", YASel, ' Data');
                ColLabel(6*aa-3)=strcat("Charging ", SelCyc(aa), " ", YASSel, ' Data');
                ColLabel(6*aa-2)=strcat("Discharging ", SelCyc(aa), " ", XASel, ' Data');
                ColLabel(6*aa-1)=strcat("Discharging ", SelCyc(aa), " ", YASel, ' Data');
                ColLabel(6*aa)=strcat("Discharging ", SelCyc(aa), " ", YASSel, ' Data');
                RawData(:,6*aa-5)=XAxisRawA(:,aa);
                RawData(:,6*aa-4)=YAxisRawA(:,aa);
                RawData(:,6*aa-3)=YAxisRawAS(:,aa);
                RawData(:,6*aa-2)=XAxisRawB(:,aa);
                RawData(:,6*aa-1)=YAxisRawB(:,aa);
                RawData(:,6*aa)=YAxisRawBS(:,aa);
            end
                bb=1;
                while bb<=R
                    if isnan(RawData(bb,:))
                        R=R-1;
                        RawData(bb,:)=[];
                    else
                        bb=bb+1;
                    end
                set(handles.RawDataTable,'Data',RawData)
                set(handles.RawDataTable,'ColumnName',ColLabel)
                end
```

```
        else
            RawData(:,1)=XAxisRaw;
            RawData(:,2)=YAxisRawA;
            RawData(:,3)=YAxisRawB;
            RawData(:,4)=YAxisRawAS;
            RawData(:,5)=YAxisRawBS;
            set(handles.RawDataTable,'Data',RawData)
        end
end
```